TRƯỜNG ĐẠI HỌC CÔNG NGHỆ SÀI GÒN

KHOA CƠ KHÍ



HƯỚNG DẪN SỬ DỤNG PROTEUS TRONG THÍ NGHIỆM VI XỬ LÝ

LỜI NÓI ĐẦU

Đây là tài liệu miễn phí dành cho Sinh viên ngành Cơ điện tử của trường STU, dùng để tham khảo và tự học mô phỏng Vi điều khiển họ 8051 sử dụng phần mềm Proteus phiên bản 7.1.

Tài liệu chỉ hướng dẫn những thao tác cơ bản nhất và một số đoạn code mẫu để sinh viên dễ dàng tham khảo, tìm hiểu. Để hiểu sâu hơn vấn đề và tăng cường kỹ năng lập trình của mình, sinh viên cần thực hiện thêm các bài tập đính kèm.

Đôi khi mô phỏng chỉ mang tính ước lượng và không sát với thực tế nên tác giả vẫn khuyến khích SV nên thử xây dựng những ứng dụng của mình trên những kit sử dụng linh kiện thực tế.

Do thời gian biên soạn ngắn nên tài liệu này không tránh khỏi những thiếu sót nhất định. Tác giả rất mong sự chia sẻ kinh nghiệm giữa những người đi trước cho những người mới.

Mọi đóng góp, xin vui lòng liên hệ:

Email: trungdphan@gmail.com

Forum: http://here.is/codientu

Mục lục

BÀI 1

LÀM QUEN VỚI PROTEUS 7.1

Để mở chương trình mô phỏng ISIS Proteus, ta thực hiện:

- **Double click** vào icon  trên desktop hoặc
- Chọn nút ***Start → Programs → Proteus 7 professional → ISIS 7 Professional***

Windows sẽ mở một cửa sổ chương trình có giao diện như sau:

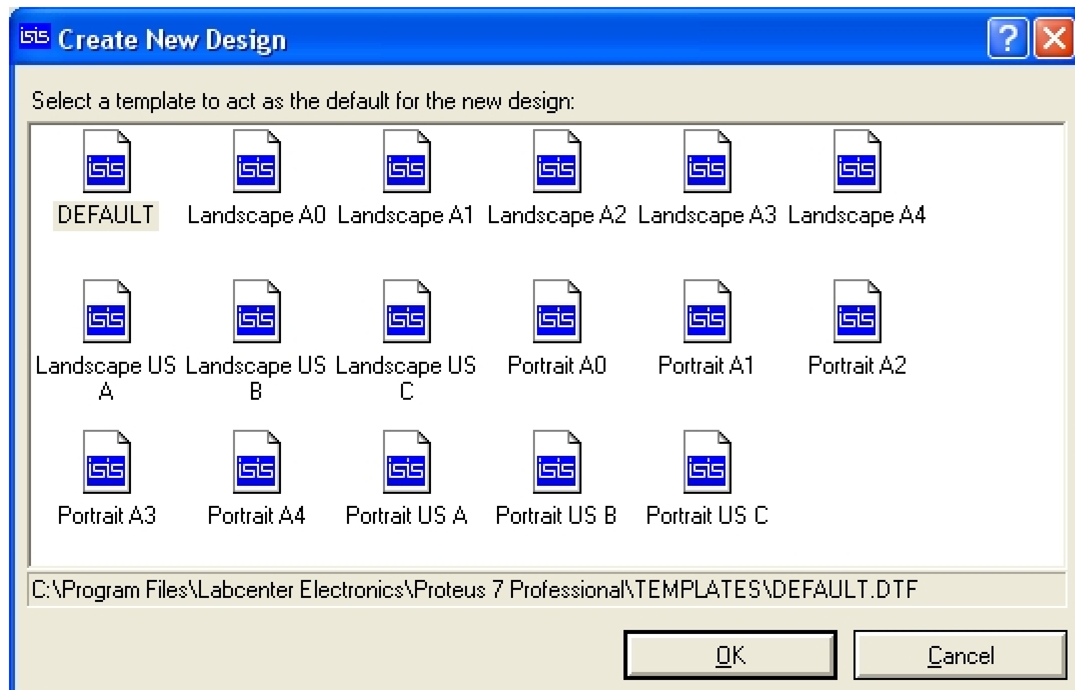
UNTITLED - ISIS Professional
File View Edit Tools Design Graph Source Debug Library Template System Help
Thanh công cụ cơ bản
Danh sách chức năng con
Thanh công cụ mô phỏng
Vùng bản vẽ
Nút thực hiện mô phỏng
Vùng báo trạng thái
No Messages
Root sheet 1
+500.0
-1600.0 th

Đây là bản vẽ mặc định của ISIS.

Tạo một bản vẽ mới bằng cách:

- Chọn menu ***File* → *New Design*** ... hoặc
- **Click** vào nút trên thanh công cụ cơ bản.

Chương trình sẽ bung ra cửa sổ hỏi về kiểu đặt bản vẽ (đặt đứng hay đặt ngang) và kích thước bản vẽ muốn tạo:



Ta chọn **DEFAULT** (không có khung tên bản vẽ) hoặc loại bản vẽ mong muốn (có khung tên).

Thay đổi kích thước của bản vẽ trong quá trình vẽ bằng cách:

- Chọn menu ***System* → *Set sheet sizes***...

Chọn khổ giấy từ A0 → A4 trong hộp thoại ***Sheet Size Configuration*** hoặc người sử dụng có thể tự nhập kích cỡ giấy cho riêng mình ở phần lựa chọn ***User***.

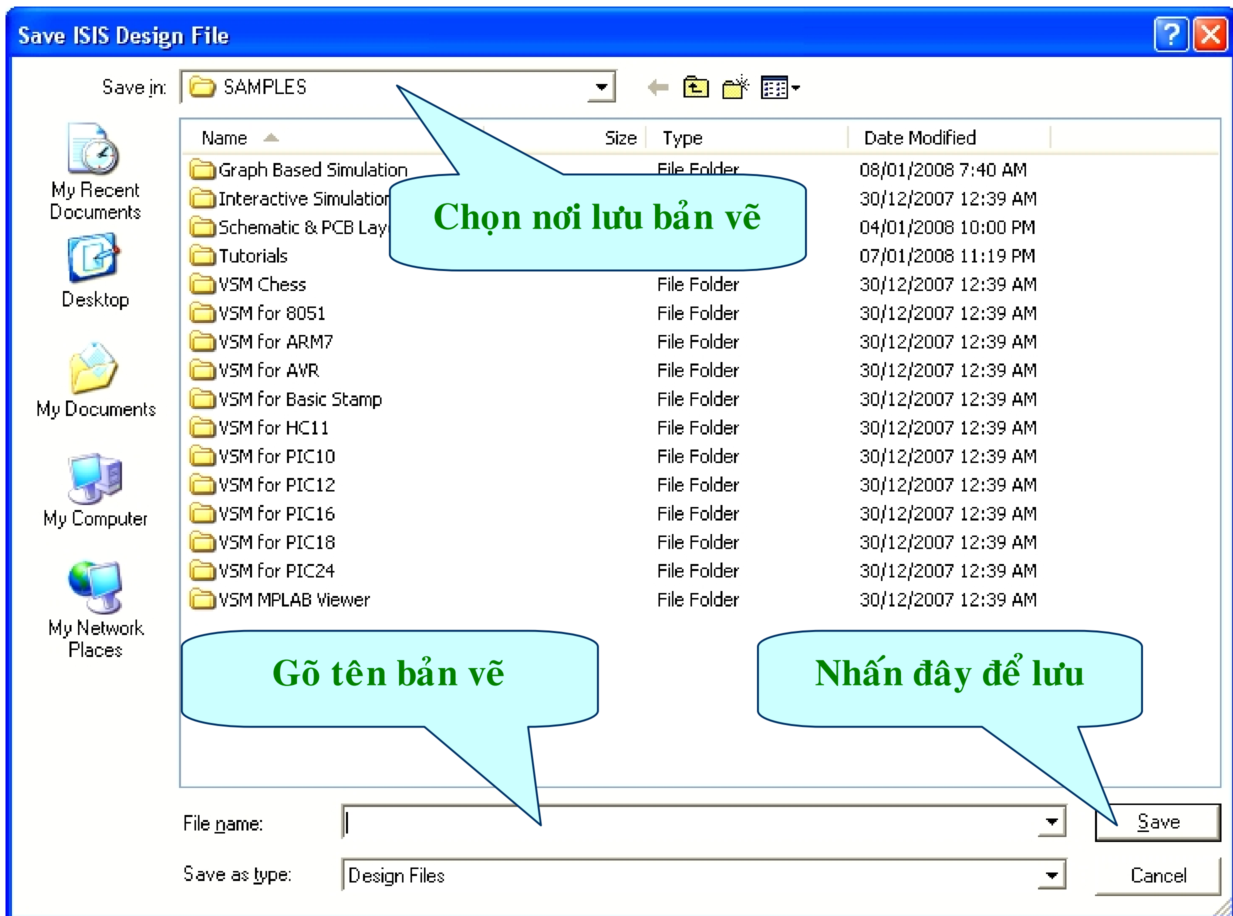
Trong quá trình vẽ mạch mô phỏng, người sử dụng nên thường xuyên lưu lại bản vẽ để tránh các trường hợp đáng tiếc xảy ra như cúp điện, treo máy,...

Để lưu một bản vẽ:

- Menu ***File* → *Save Design*** hoặc
- Menu ***File* → *Save Design As***... (lưu bản vẽ này với tên khác) hoặc
- **Click** vào nút hoặc
- Nhấn **CTRL + S**

Nếu bản vẽ chưa lưu hoặc được lưu với tên khác, hộp thoại ***Save ISIS Design File*** sẽ xuất hiện. Chọn nơi sẽ lưu bản vẽ trong danh sách ***Save in***, gõ tên bản vẽ ở phần ***File name***, nhấn nút ***Save***.

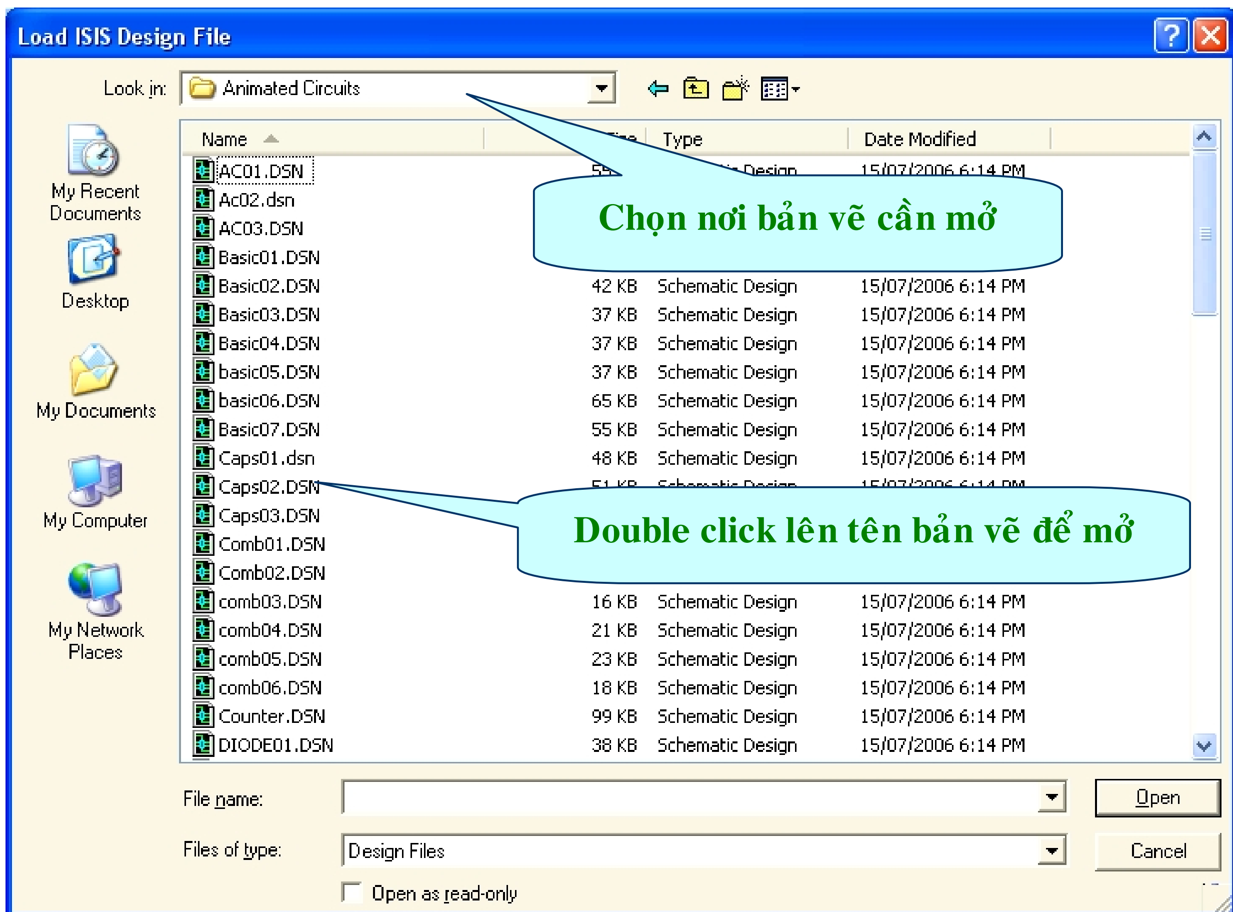
Bản vẽ hiện hành sẽ được lưu với phần mở rộng là **DSN**



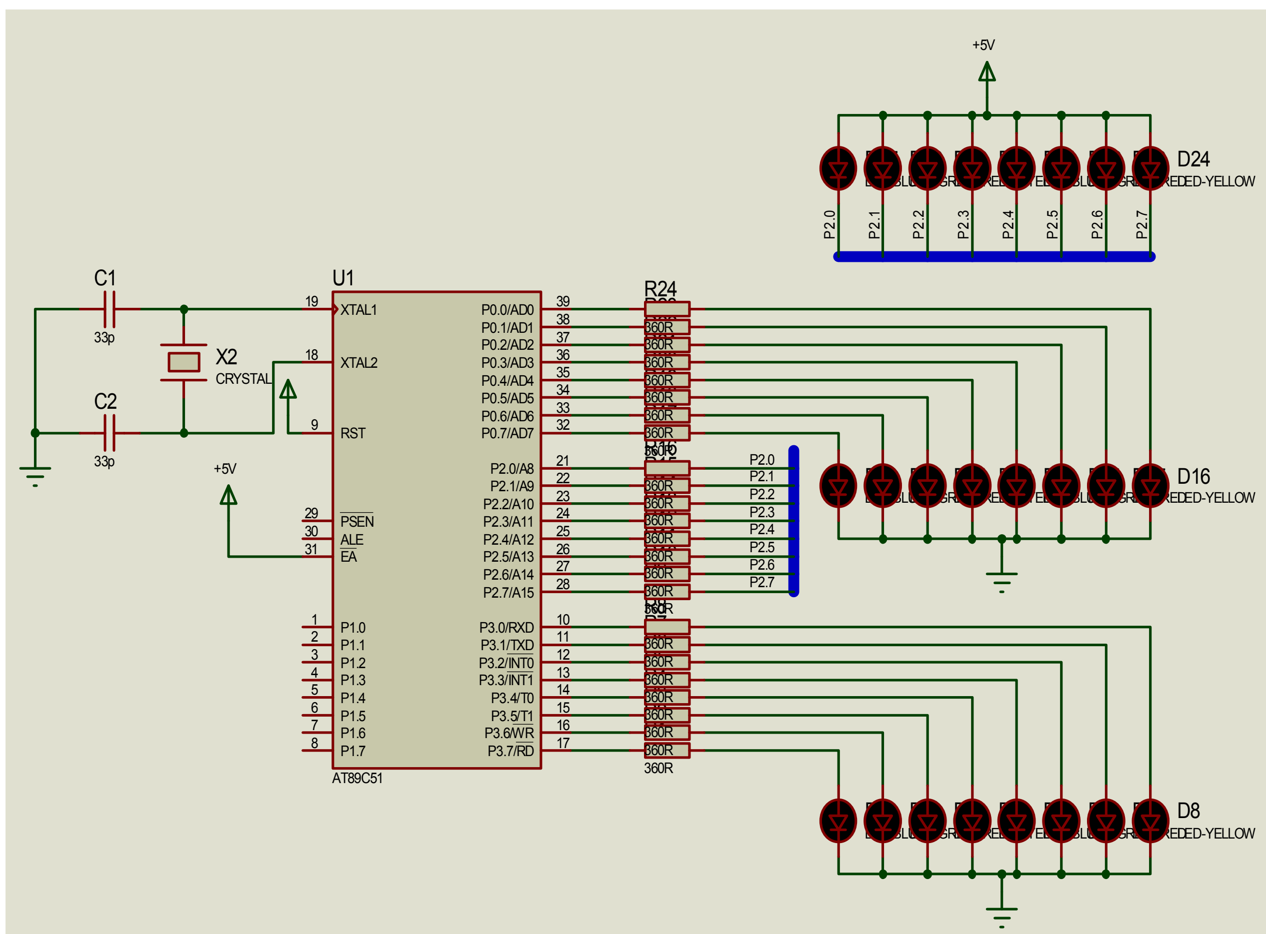
Để mở lại bản vẽ đã lưu, chọn:

- Menu ***File*** **→** ***Open Design…*** hoặc
- **Click** vào nút hoặc
- nhấn **CTRL + O**

Hộp thoại ***Load ISIS Design File*** sẽ xuất hiện. Chọn nơi bản vẽ cần mở được lưu trữ trong phần ***Look in*** và **double click** lên tên bản vẽ muốn mở.



Bây giờ ta sẽ dùng phần mềm Proteus 7.1 để vẽ mạch mô phỏng sau:

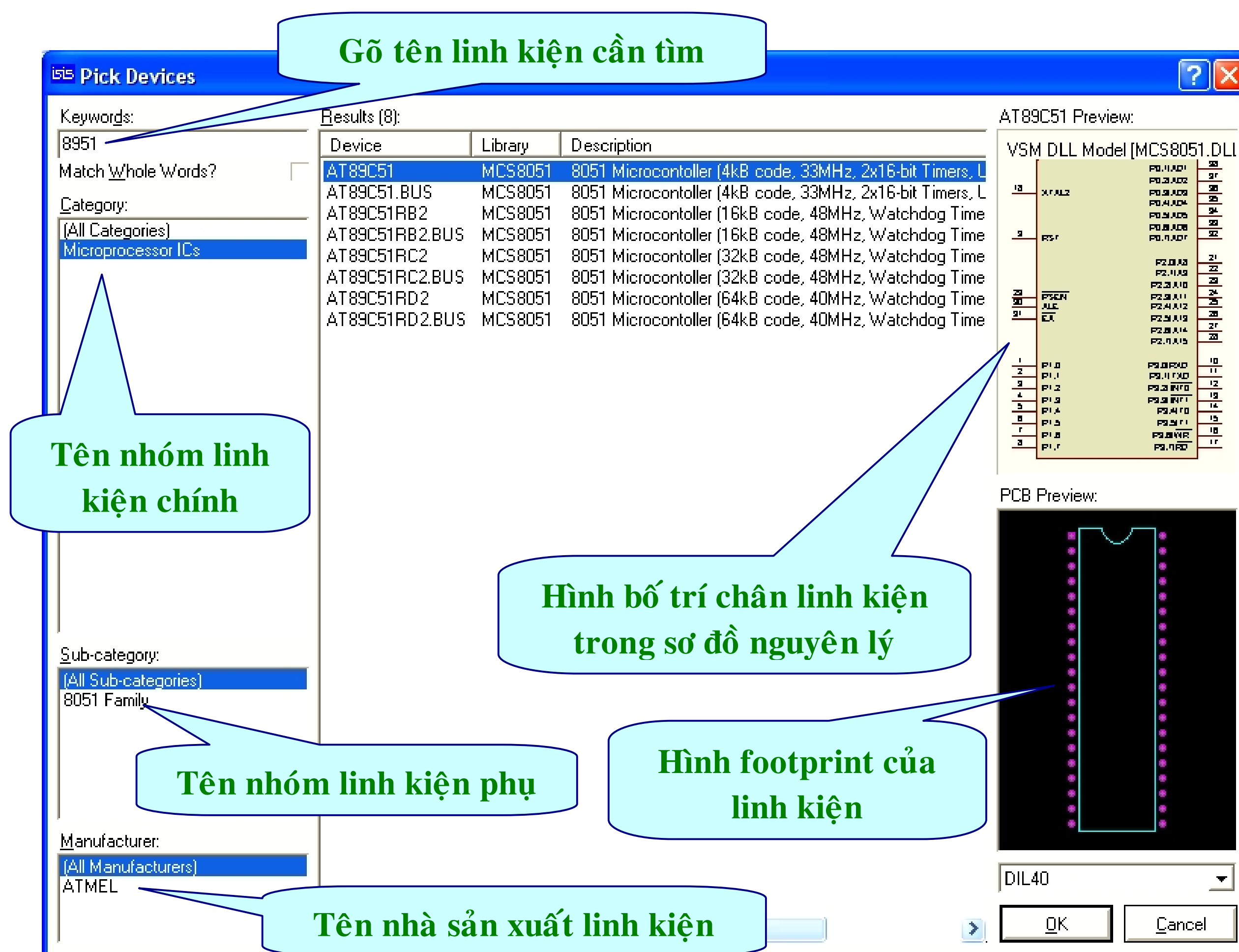


Mở chương trình Proteus. Để đặt linh kiện lên bản vẽ, ta cần lấy linh kiện từ thư viện linh kiện ra.

Lấy linh kiện từ thư viện linh kiện:

- Chọn menu ***Library*** **➔** ***Pick device/symbol…*** hoặc
- **click** vào biểu tượng hoặc
- **click** vào chữ P trên P L DEVICES hoặc
- nhấn phím **p** trên bàn phím

Chương trình sẽ mở ra cửa sổ ***Pick Devices*** như sau:



Ta gõ tên linh kiện cần tìm ở khung ***Keywords***, tên này không cần gõ chính xác mà chỉ nên gõ phần đặc trưng của tên linh kiện. Chương trình sẽ hiển thị kết quả tìm kiếm ở khung ***Results*** và ***Category***. Nếu không nhớ (hoặc không biết tên linh kiện) ta có thể tìm kiếm linh kiện thông qua nhóm linh kiện (***Category***), nhóm con của linh kiện (***Sub-category***), tên nhà sản xuất (***Manufacturer***) và các linh kiện được hiển thị trong mục ***Results***.

Khi chọn một linh kiện bất kỳ trong khung ***Results***, chương trình sẽ hiển thị hình bố trí chân của linh kiện trong sơ đồ nguyên lý ở khung ***Preview*** và dạng chân mạch in của linh kiện ở khung ***PCB Preview***.

Chú ý: chương trình Proteus **KHÔNG** cho phép thay đổi vị trí bố trí chân linh kiện trong sơ đồ nguyên lý nên người sử dụng phải chọn đúng dạng bố trí chân mong muốn theo yêu cầu mô phỏng.

Nhấn **double click** lên tên linh kiện để lấy linh kiện đặt vào danh sách sử dụng. Lặp lại quá trình này đến khi lấy đủ linh kiện và kết thúc bằng cách **click** vào nút ***OK*** để dừng.

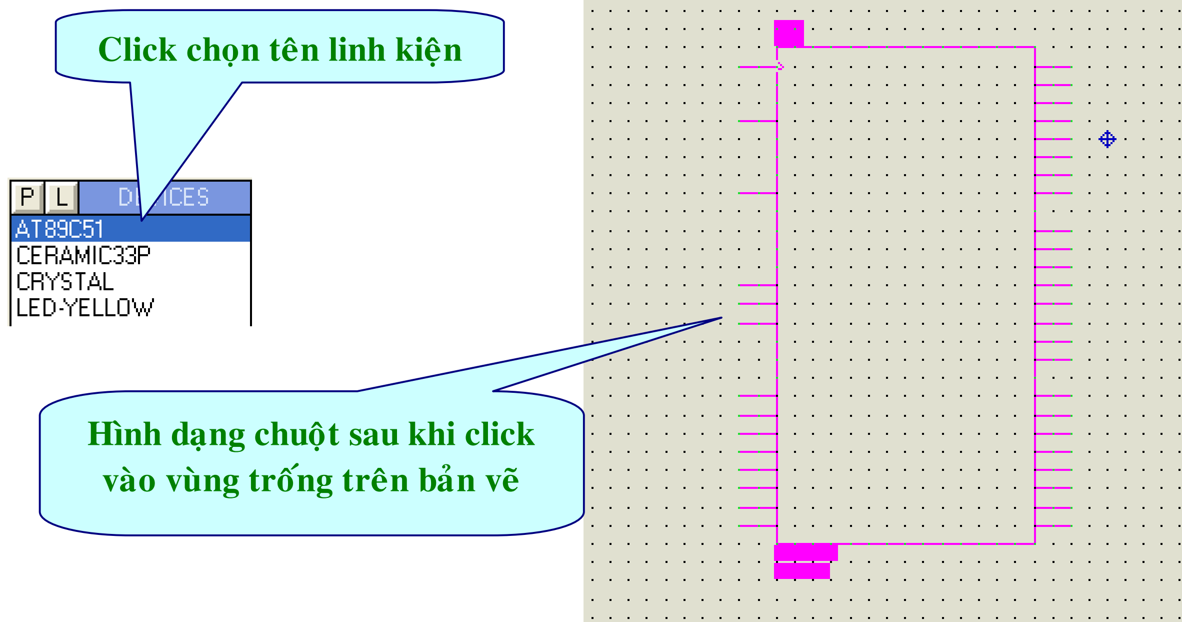
Để vẽ được mạch trên, ta cần lấy đầy đủ các linh kiện có trong danh sách bên

P L DEVICES
AT89C51
CERAMIC33P
CRYSTAL
LED-YELLOW
MINRES330R

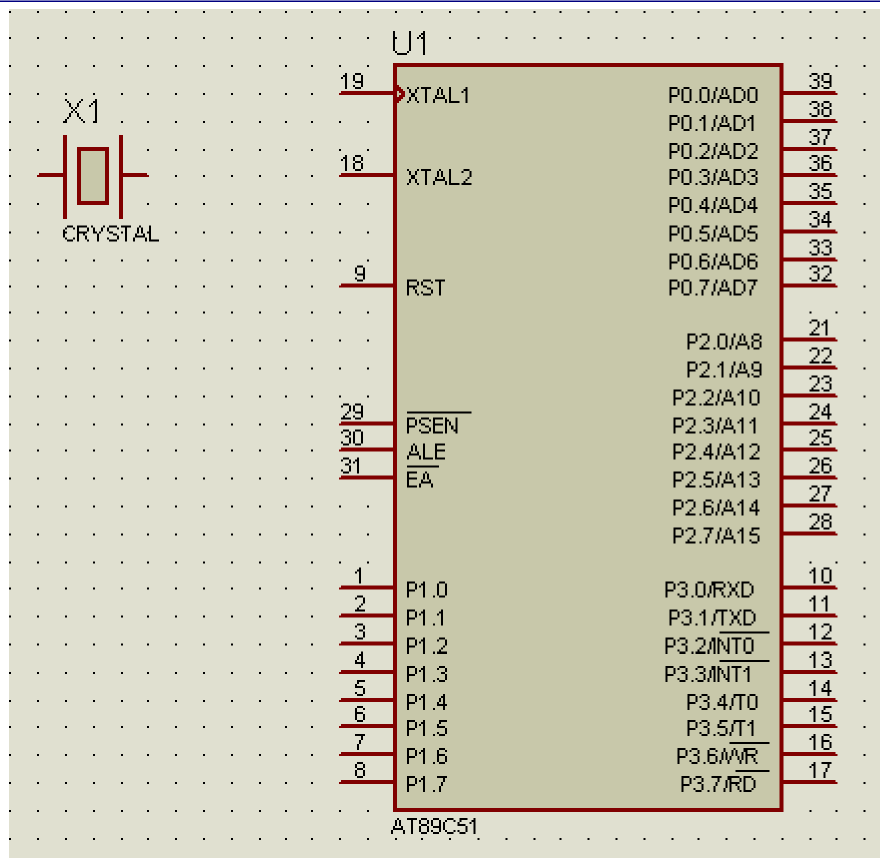
Đặt linh kiện lên bản vẽ:

1. **click** chọn nút (***component mode***)
2. **click** chọn tên linh kiện trong danh sách. Tên linh kiện được chọn sẽ được tô màu xanh và hình dạng linh kiện sẽ hiện ở khung phía trên danh sách.
3. **click** vào vùng trống của bản vẽ, hình linh kiện sẽ thay thế con trỏ
4. **click** vào nơi muốn đặt linh kiện để kết thúc việc đặt linh kiện.

Lặp lại các bước **2→ 4** để đặt các linh kiện lên bảng vẽ. Nếu đặt các linh kiện cùng tên thì chỉ cần lặp bước **3 → 4**.



Dùng thao tác trên để hoàn thành phần đặt linh kiện như hình sau.



Trong quá trình đặt linh kiện, đôi khi ta muốn đặt linh kiện nằm ngang hoặc thẳng đứng, ta thực hiện thao tác xoay linh kiện như sau:

Xoay linh kiện chưa đặt trên bản vẽ:



1. **Click** chọn tên linh kiện muốn xoay trong danh sách linh kiện.
2. **Click** chọn thao tác xoay tương ứng trên thanh công cụ (hình bên)

   Nút - thực hiện xoay linh kiện cùng chiều kim đồng hồ.

   Nút - thực hiện xoay linh kiện ngược chiều kim đồng hồ.

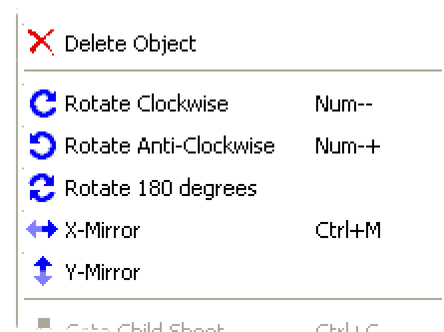
   Nút - thực hiện lấy đối xứng ngang linh kiện.

   Nút - thực hiện lấy đối xứng dọc linh kiện.

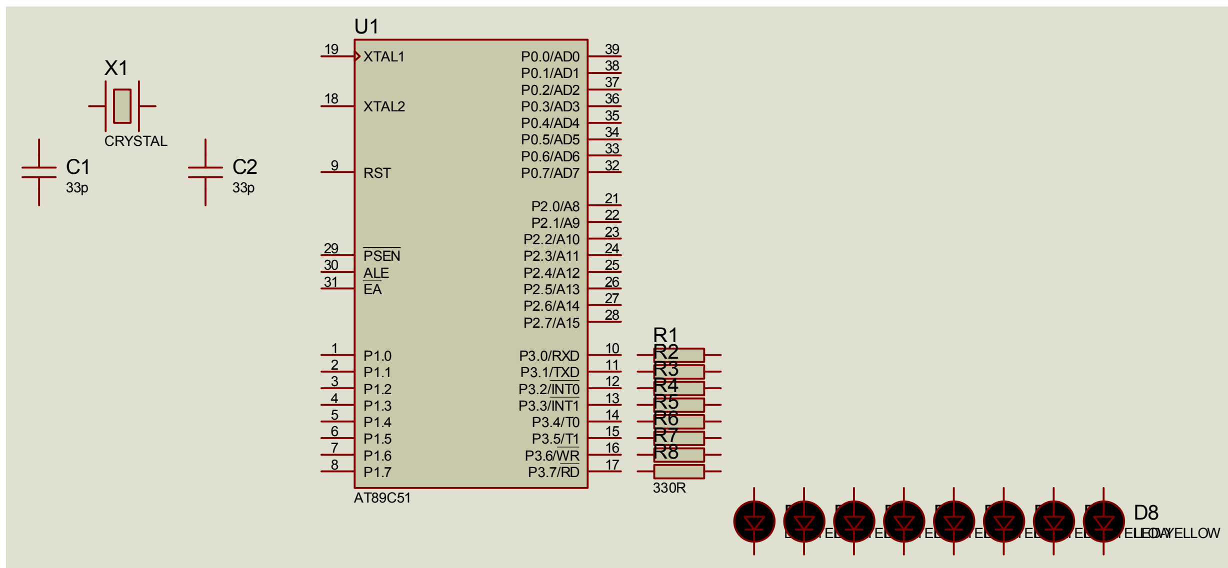
   Ô 180° - chỉ cho phép nhập các góc quay là 0^O, 90^O, 180^O, 270^O, 360^O.

Xoay linh kiện đã đặt trên bản vẽ:

1. **click phải** chuột lên **trên** linh kiện muốn xoay.
2. chọn thao tác xoay tương ứng trong menu popup



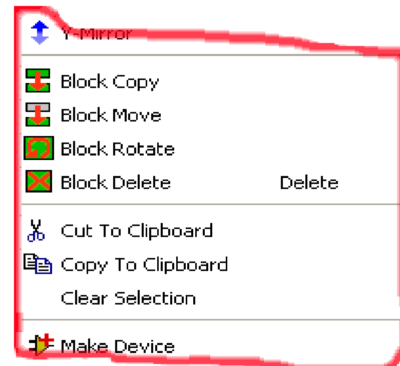
Bây giờ ta đặt thêm 2 tụ và các linh kiện khác để có được mạch như hình vẽ sau.



Nếu dùng phương pháp đặt linh kiện ở trên để đặt hết led và điện trở sẽ rất lâu và không hiệu quả, ta nên dùng phương pháp sao chép các phần mạch giống nhau.

Sao chép/ di chuyển/ quay/ xóa các đối tượng:

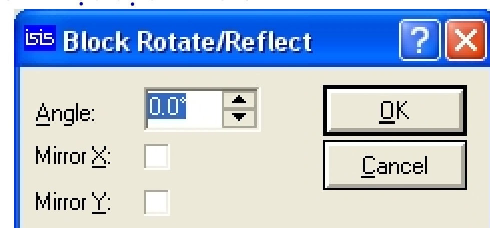
- **Nhấn và giữ chuột** trái (hoặc phải đều được), **kéo rê chuột** để vẽ nên đường bao hình chữ nhật bao quanh nhóm đối cần sao chép (đối tượng được chọn sẽ chuyển sang màu đỏ).
- Chọn thao tác trên nhóm đối tượng vừa chọn **từ thanh công cụ cơ bản** ( ) hoặc
- **Click** phải **bên trong vùng chọn** đối tượng, chọn thao tác mong muốn.



Nút ***Block Copy*** - **sao chép** đối tượng **có đổi tên tham chiếu**.

Nút ***Block Move*** - **di chuyển** đối tượng đến một vị trí mới.

Nút ***Block Rotate*** - **xoay** đối tượng với góc chỉ định trong hộp thoại ***Block Rotate/Reflect*** (tương tự phần xoay linh kiện ở trên)



Nút ***Block Delete*** - **xóa** đối tượng được chọn.

Nút ***Cut To Clipboard*** - **cắt** đối tượng được chọn ra khỏi bản vẽ và lưu một bản sao trong bộ nhớ.

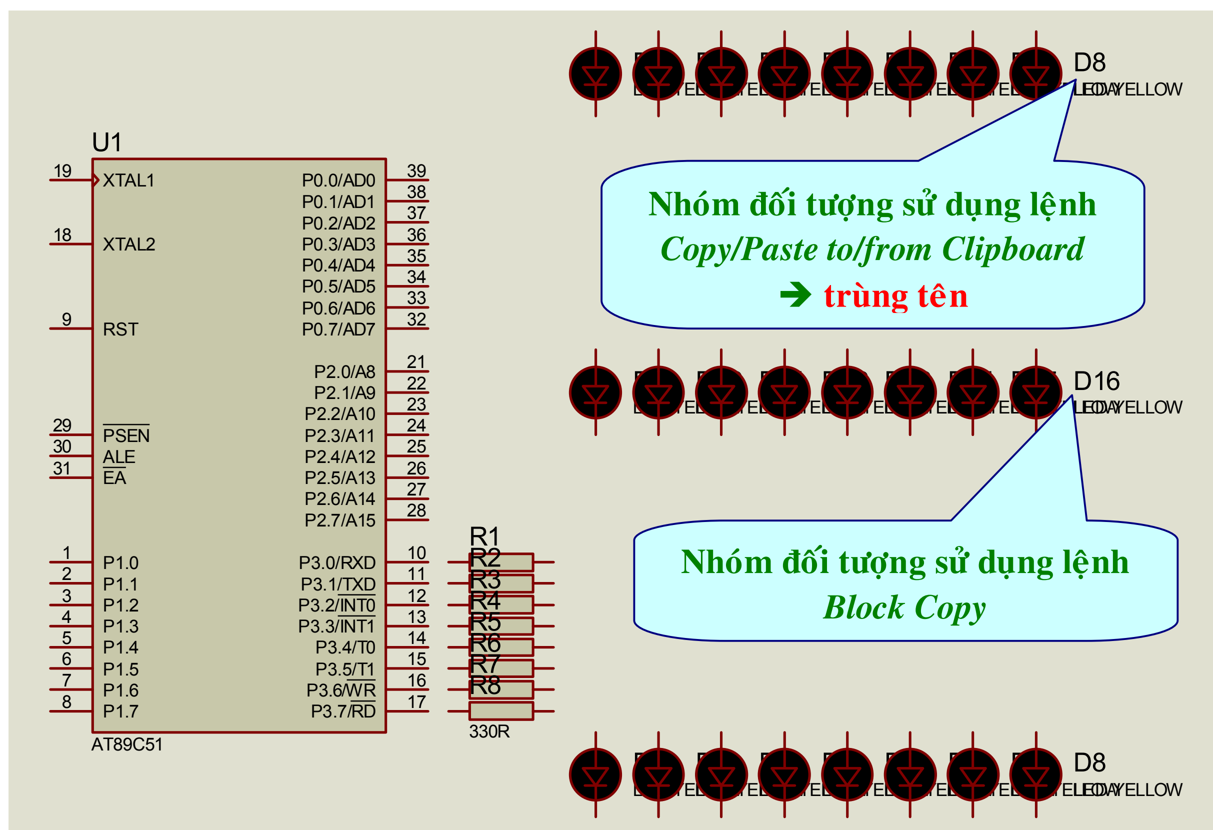
Nút ***Copy To Clipboard*** - **sao chép** đối tượng được chọn và lưu một bản sao trong bộ nhớ.

Nút ***Paste From Clipboard*** - **dán một** đối tượng đã sao chép trong bộ nhớ ra bản vẽ.

Nút ***Clear Selection*** – **huỷ** lựa chọn đối tượng (đối tượng được chọn sẽ trở lại màu mặc định)

Hành động của 2 nhóm nút ***Block Copy*** và ***Cut/Copy/Paste To/From Clipboard*** có sự khác nhau là:

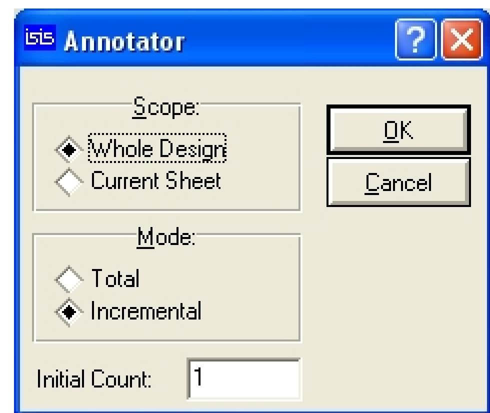
- nút ***Block Copy*** sẽ thay đổi tên tham chiếu của đối tượng.
- Nhóm nút còn lại sẽ không đổi tên tham chiếu của đối tượng → điều này sẽ phát sinh lỗi "**có nhiều linh kiện trùng tên**" khi thực hiện mô phỏng.



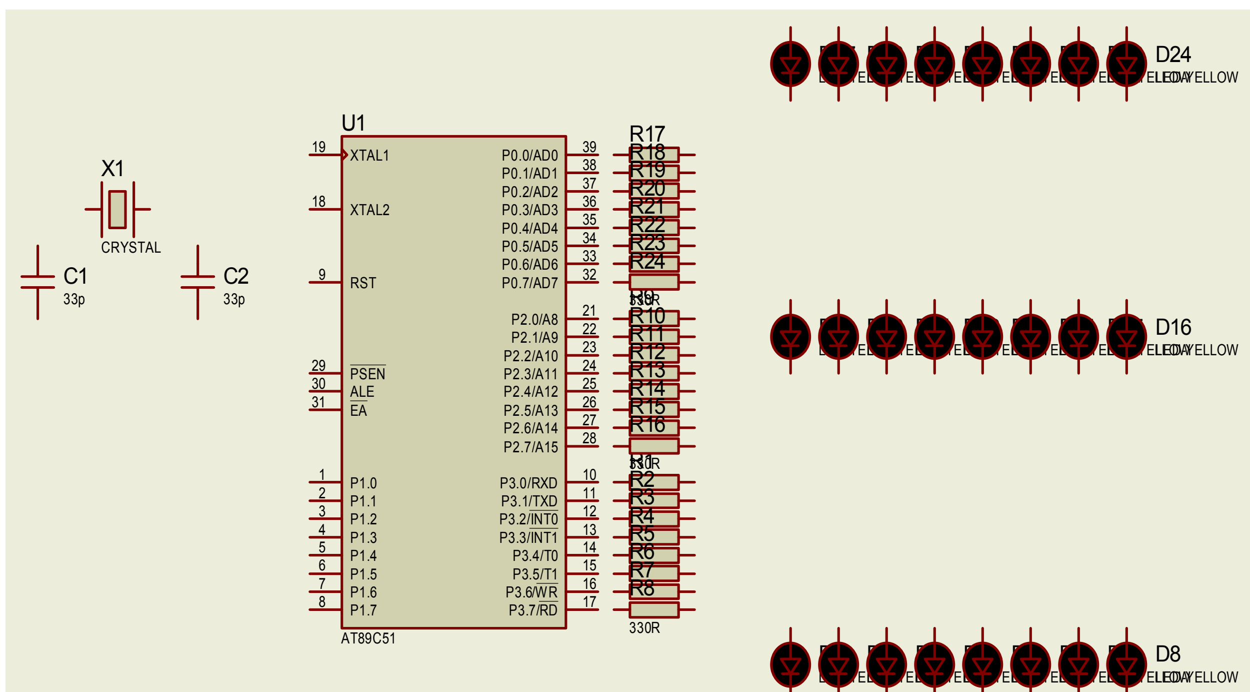
Có 2 cách để khắc phục lỗi trùng tên:

1. Sử dụng thao tác ***Block Copy*** để sao chép các nhóm mạch giống nhau.
2. Dùng lệnh đổi tên tham chiếu:
   - Chọn menu ***Tools → Global Annotator…*** , hộp thoại ***Annotator*** sẽ bung ra.
   - Chọn ***Whole Design*** trong ***Scope***.

- Chọn ***Total*** trong ***Mode***
- Nhấn **OK** để xác nhận công việc.



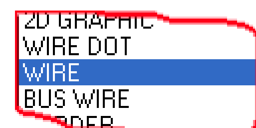
Tiếp tục đặt linh kiện, sao chép các phần mạch giống nhau, ta sẽ thu được phần mạch như sau:



Bây giờ ta bắt đầu vẽ các đường mạch nối các chân linh kiện với nhau.

Để vẽ đường mạch:

- Chọn nút ***2D graphic line mode*** trên thanh công cụ mô phỏng.
- Chọn ***WIRE*** trong danh sách.

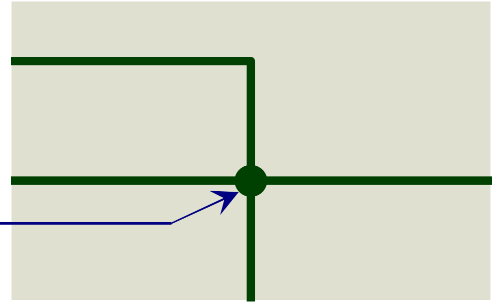


- **Click** vào một chân linh kiện để bắt đầu vẽ đường mạch và **click** vào chân linh kiện khác để kết thúc (nếu dây nối chỉ có một đầu nối vào chân linh kiện thì **double click** để kết thúc nối dây)

Chương trình Proteus sẽ tự động chọn công cụ nối dây cho mạch khi người sử dụng đưa con trỏ đến chân linh kiện muốn nối dây (chân linh kiện sẽ xuất hiện dấu vuông đỏ khi cho phép nối dây).

Chú ý: hai đường mạch chỉ được xem là cắt nhau khi có dấu tròn màu xanh lá tại điểm giao của hai đường mạch.

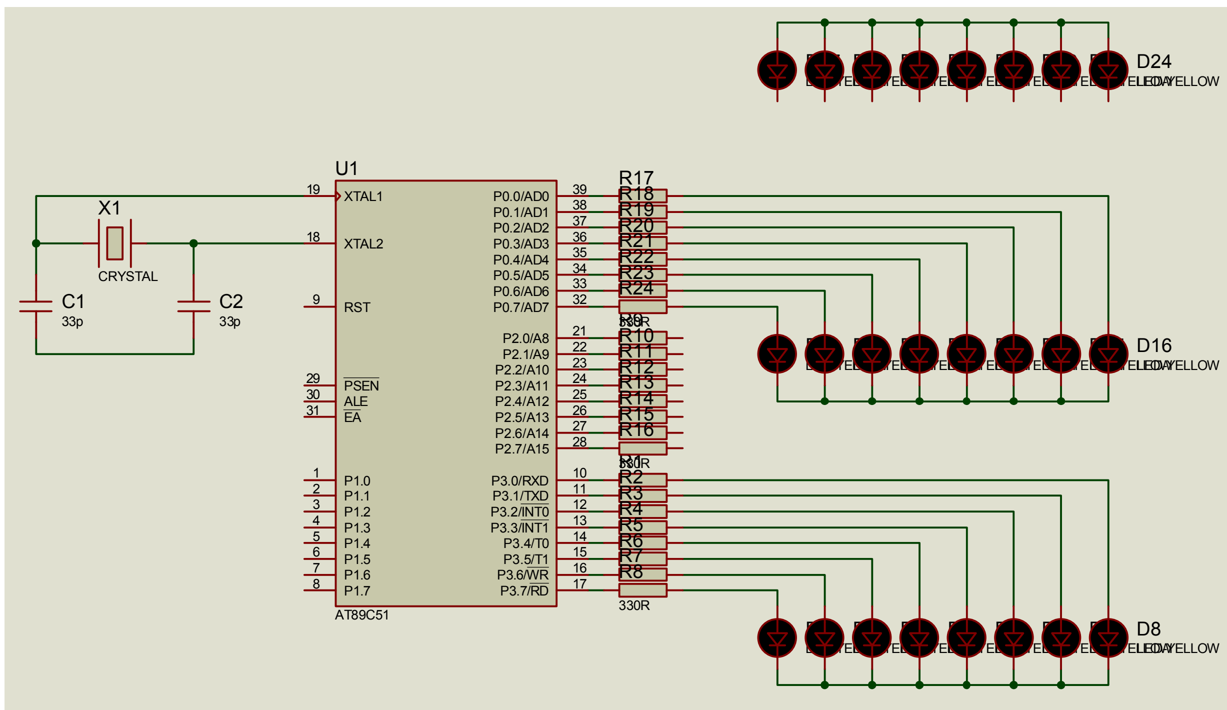
Điểm giao 2 đường mạch



Để xóa đường mạch:

- **Click phải 2 lần** lên một đường mạch muốn xóa (cách này cũng có thể dùng để xóa linh kiện).

Sau khi thực hiện vẽ kết nối mạch ta được hình dạng như sau

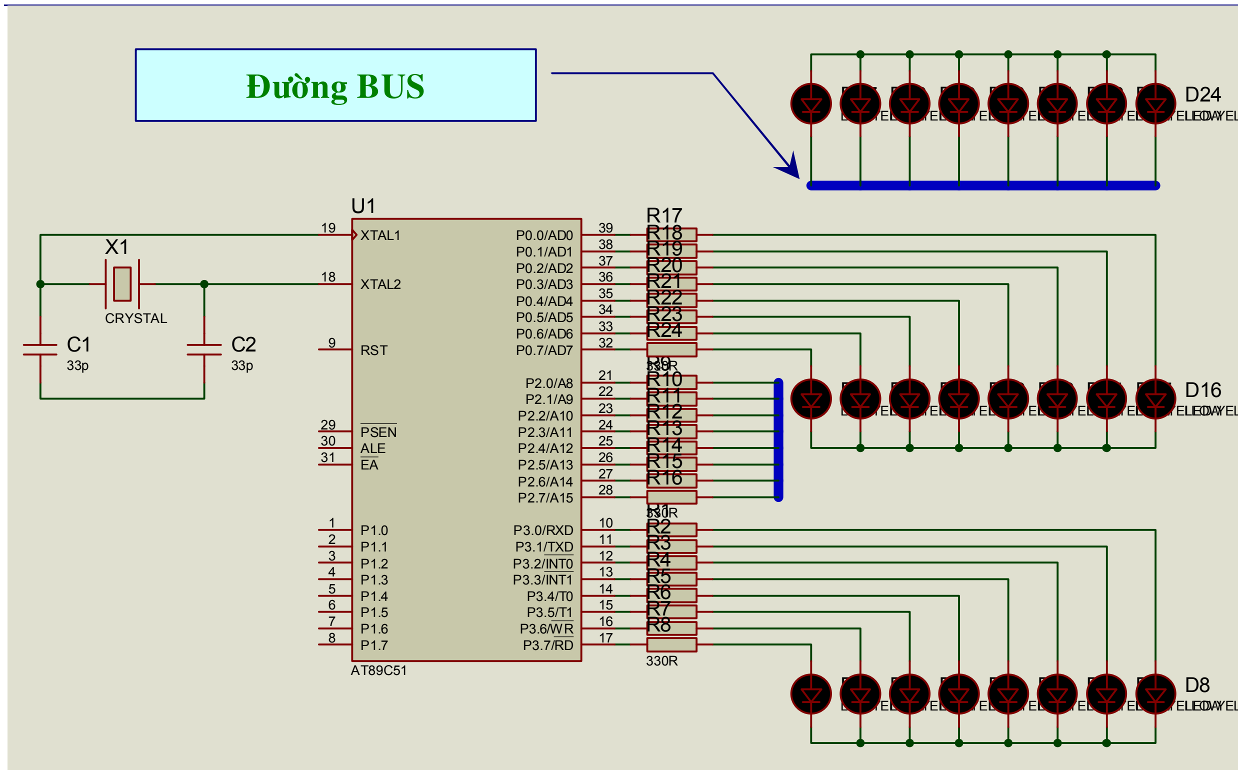


Để vẽ đường BUS (đường mạch màu xanh dương):

- **Click** chọn nút ***Buses mode*** ở thanh công cụ mô phỏng**.**
- **Nhấn và thả chuột trái** lên vùng trống trên bản vẽ, nơi cần vẽ đường BUS.
- **Rê chuột** để vẽ đường BUS (**click** để bẻ góc đường BUS đang vẽ).
- **Double click** để kết thúc vẽ đường BUS.

Để xóa đường BUS: thực hiện tương tự như xóa đường mạch.

Sau khi thực hiện vẽ BUS và kết nối các đường mạch vào BUS ta được dạng mạch như sau:

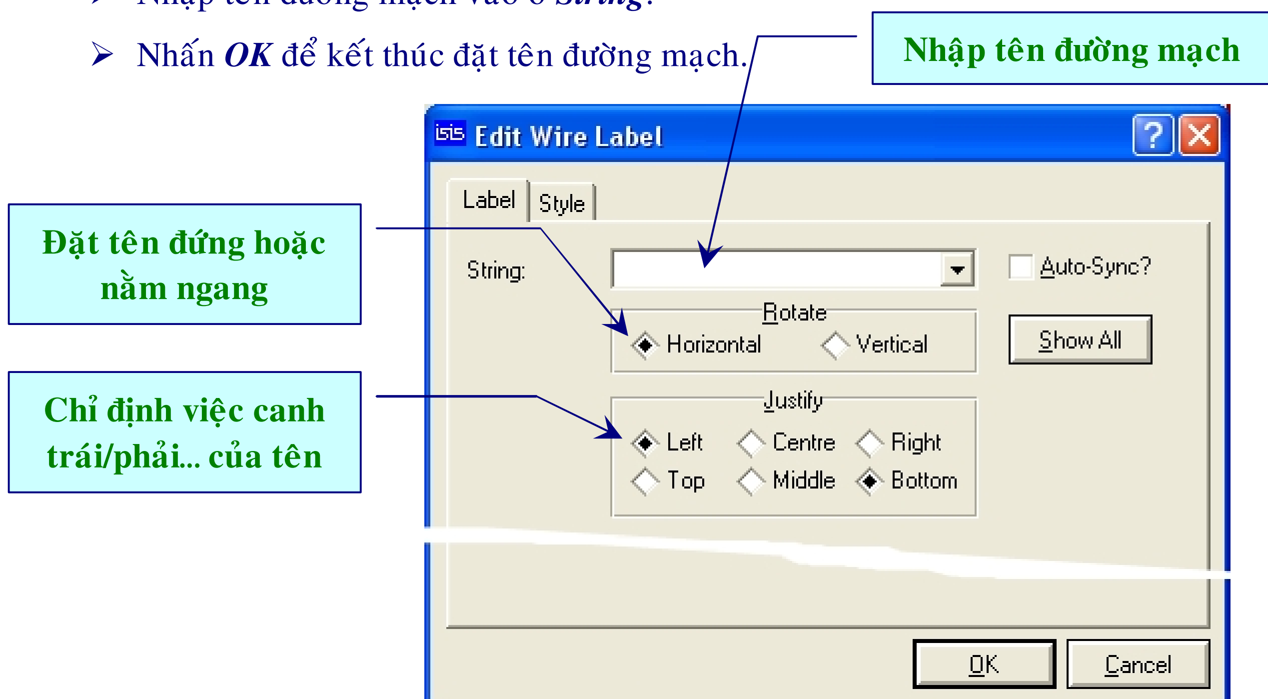


Khi sử dụng đường BUS, ta phải đặt tên cho các đường mạch được kết nối vào BUS theo nguyên tắc:

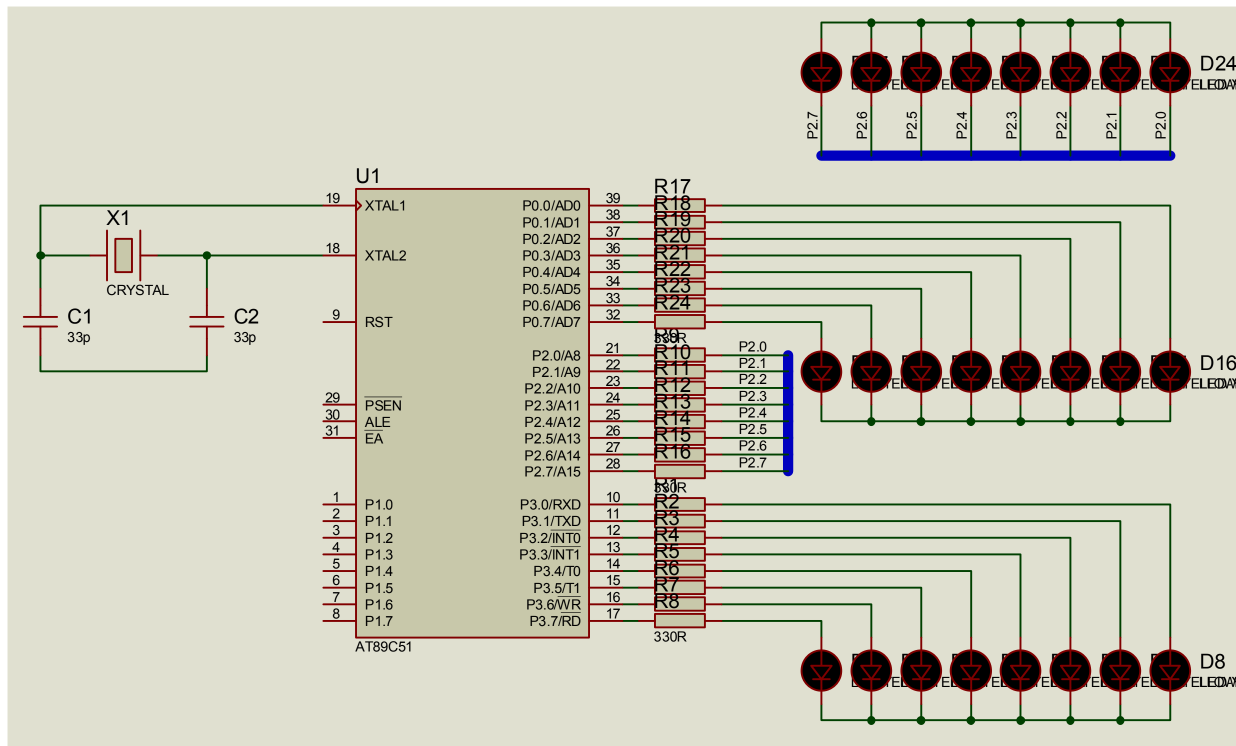
"Các đường mạch cùng tên sẽ được nối lại với nhau"

Để đặt tên cho các đường mạch:

- Chọn nút ***Wire label mode*** LBL trên thanh công cụ mô phỏng.
- **Click** vào đường mạch muốn đặt tên, chương trình sẽ bật hộp thoại ***Edit Wire Label***.
- Nhập tên đường mạch vào ô ***String***.
- Nhấn ***OK*** để kết thúc đặt tên đường mạch.



Sau khi thực hiện đặt tên cho đường BUS, ta sẽ được bản vẽ mạch như sau:



Bây giờ ta sẽ đặt nguồn và mass (đất hay GROUND) cho mạch.

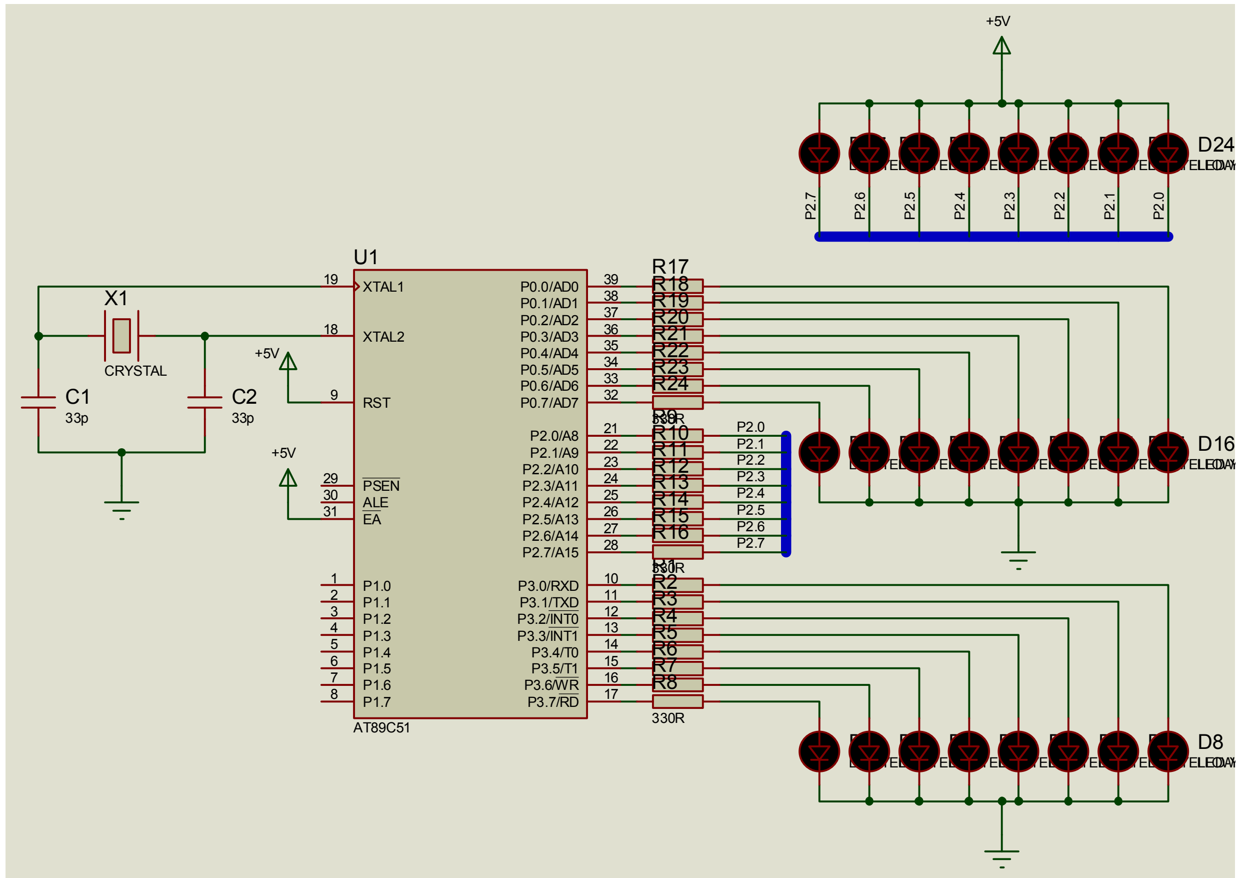
Đặt nguồn và đất:

- **Click** chọn nút ***Inter-sheet terminals*** ( ) trên thanh công cụ mô phỏng, cửa sổ P TERMINALS sẽ mở ra cho phép người dùng lựa chọn loại terminal cần dùng. Có hai loại terminal chúng ta có thể sử dụng ở đây là:
  - ***GROUND***: vẽ ký hiệu mass (ground), cung cấp điện áp 0 V cho mô phỏng mạch.
  - ***POWER***: vẽ ký hiệu nguồn dương hoặc âm.
- **Click** vào vùng trống trên bản vẽ để đặt nguồn lên bản vẽ.
- **Click phải** để kết thúc việc đặt nguồn.
- **Double click** lên ký hiệu nguồn vừa đặt trên bản vẽ(hoặc **click** phải lên ký hiệu nguồn, chọn ***Edit Properties***), nhập giá trị nguồn điện vào ô ***String***. Ví dụ: +5V.

Chú ý:

- Phải đặt chính xác nguồn dương hoặc âm, không được để trống như sau: **5V**
- Nếu không ghi rõ là nguồn bao nhiêu thì chương trình sẽ lấy giá trị mặc định tuỳ theo mô phỏng.
- Các IC không có chân nguồn và mass thì chương trình đã mặc định cấp nguồn cho IC rồi.

Hoàn tất bản vẽ mạch ta được dạng như sau:



Ta nên lưu lại bản vẽ này trước khi tiến hành các bước kế tiếp.

Trong quá trình vẽ, ta có thể sử dụng thêm các công cụ phóng to thu nhỏ để việc vẽ dễ dàng hơn.

Phóng to/thu nhỏ bản vẽ – chuyển tâm xem bản vẽ:

Chọn nút chức năng tương ứng sau ở thanh công cụ cơ bản:

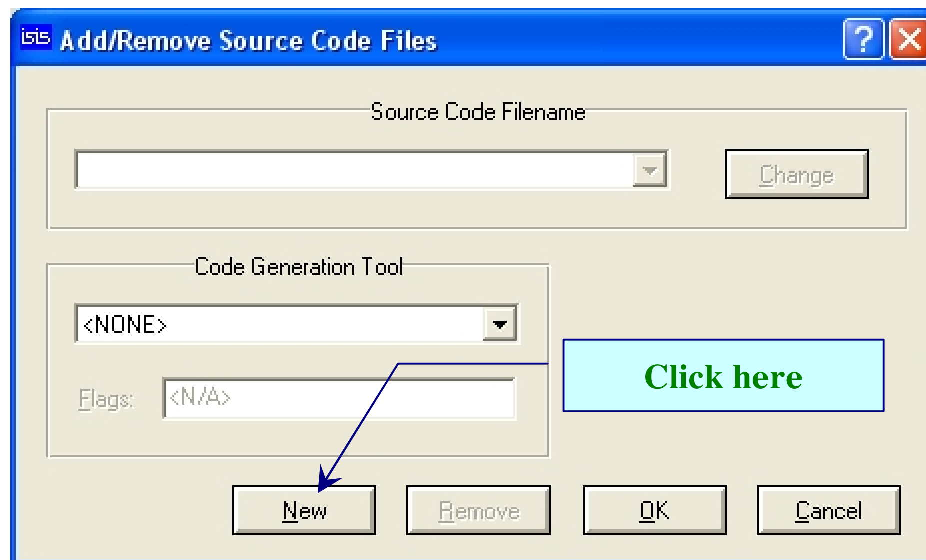
- Nút ***Center At Cursor*** - chuyển tâm xem bản vẽ.
- Nút ***Zoom In*** – phóng to bản vẽ để xem.
- Nút ***Zoom Out*** – thu nhỏ bản vẽ đang xem.
- Nút ***Zoom To View Entire Sheet*** – cho phép xem toàn bộ bản vẽ.
- Nút ***Zoom To Area*** – nút phóng to một vùng được chọn. Chọn vùng tương tự chọn một nhóm các đối tượng.

Viết chương trình vi điều khiển:

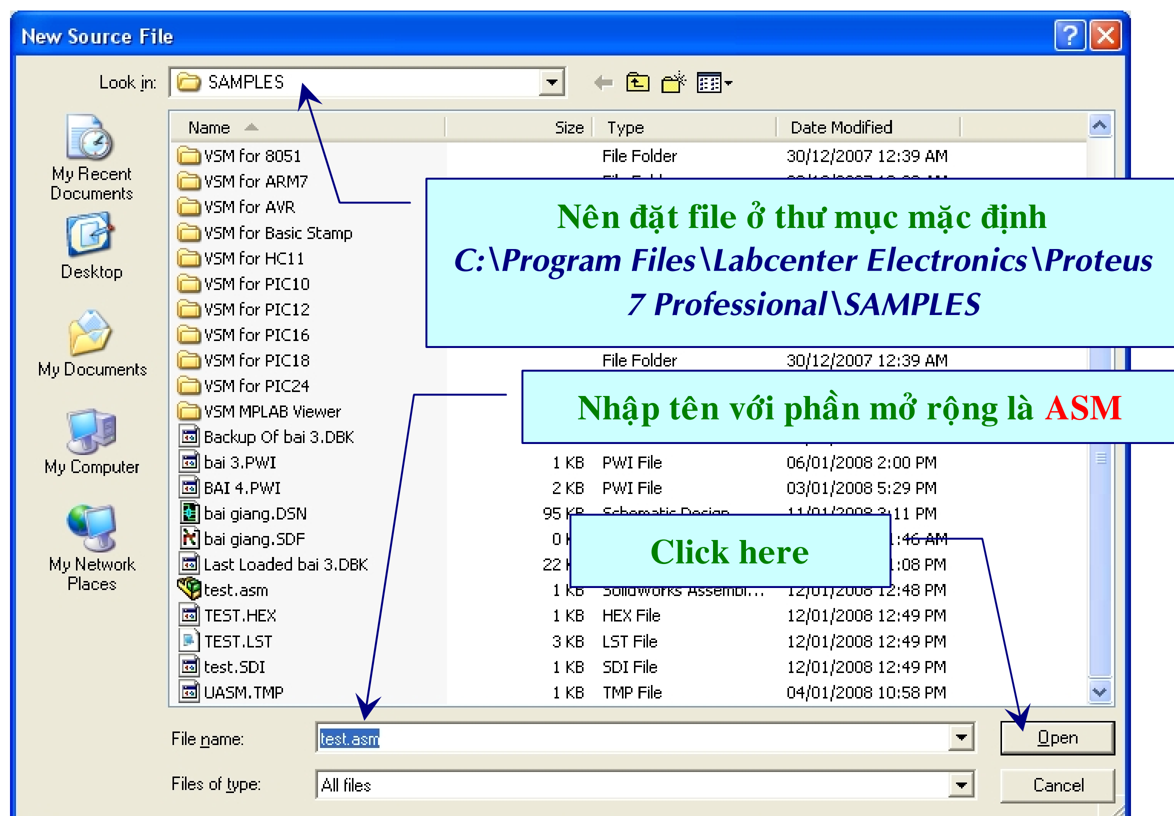
Bây giờ ta thực hiện viết chương trình để thực hiện mô phỏng cho vi điều khiển họ 8051. Nếu ta dùng một phần mềm biên dịch của hãng khác như Keil, MIDE-51,... thì ta biên dịch chương trình vi điều khiển bằng phần mềm đó ra file ***.HEX** rồi thực hiện bước tiếp theo (bỏ qua bước này).

Nếu dùng trình biên dịch của Proteus 7.1 thì ta thực hiện các bước sau:

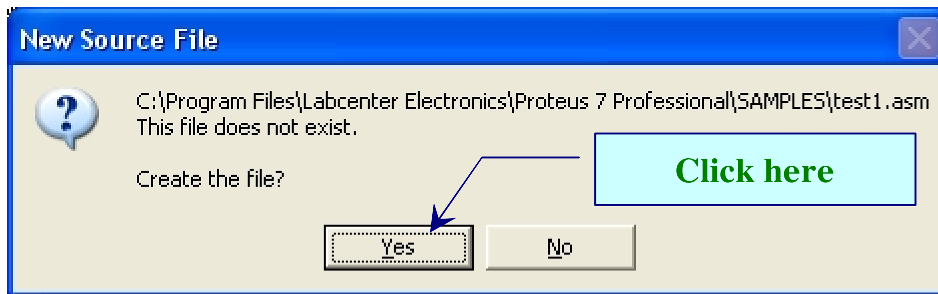
- Chọn menu ***Source → Add/Remove Source File...***
- Hộp thoại ***Add/Remove Source Code Files*** sẽ xuất hiện, **click** chọn nút ***New***



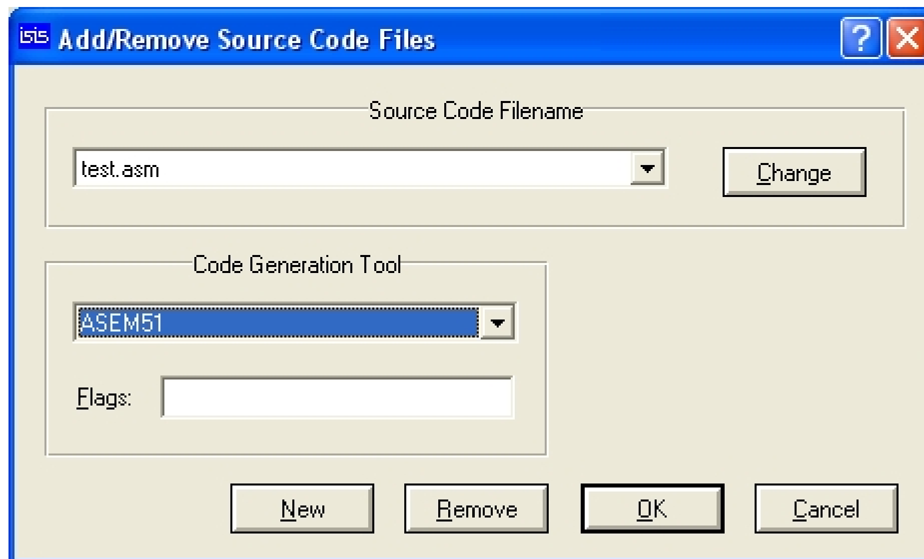
- Nhập tên file vào ô ***File name***, nên nhập thêm phần mở rộng của file là **ASM** (như hình)



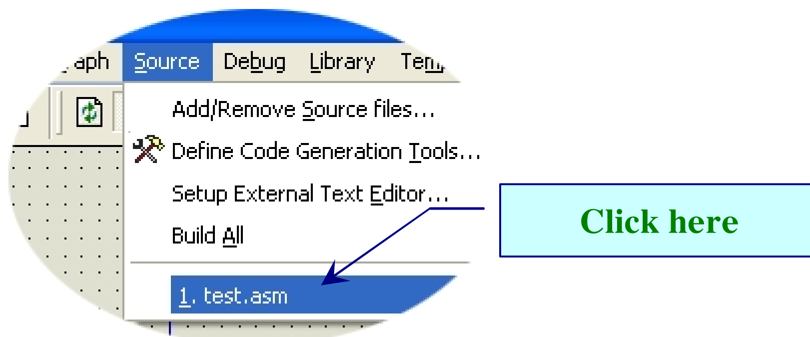
- **Click** vào nút ***Open*** và trả lời ***Yes*** để tạo file mới trong hộp thoại ***New Source File***.



- Chọn ***ASEM51*** trong phần ***Code Generation Tool*** rồi nhấn ***OK***.



- Chọn menu ***Source → test.asm*** vừa tạo. Cửa sổ ***Source Editor*** sẽ xuất hiện, đây là nơi dùng để nhập chương trình.

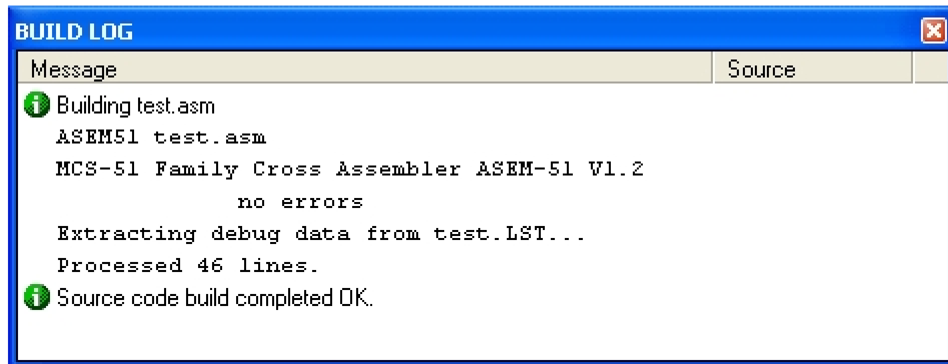


- Trong cửa sổ ***Source Editor***, hãy gõ đoạn chương trình sau:

```
MAIN:
        MOV  P3,#11110000B
        JMP  $
END
```

- Chọn menu ***File → Save*** để lưu lại chương trình vi điều khiển.
- Đóng cửa sổ ***Source Editor.***

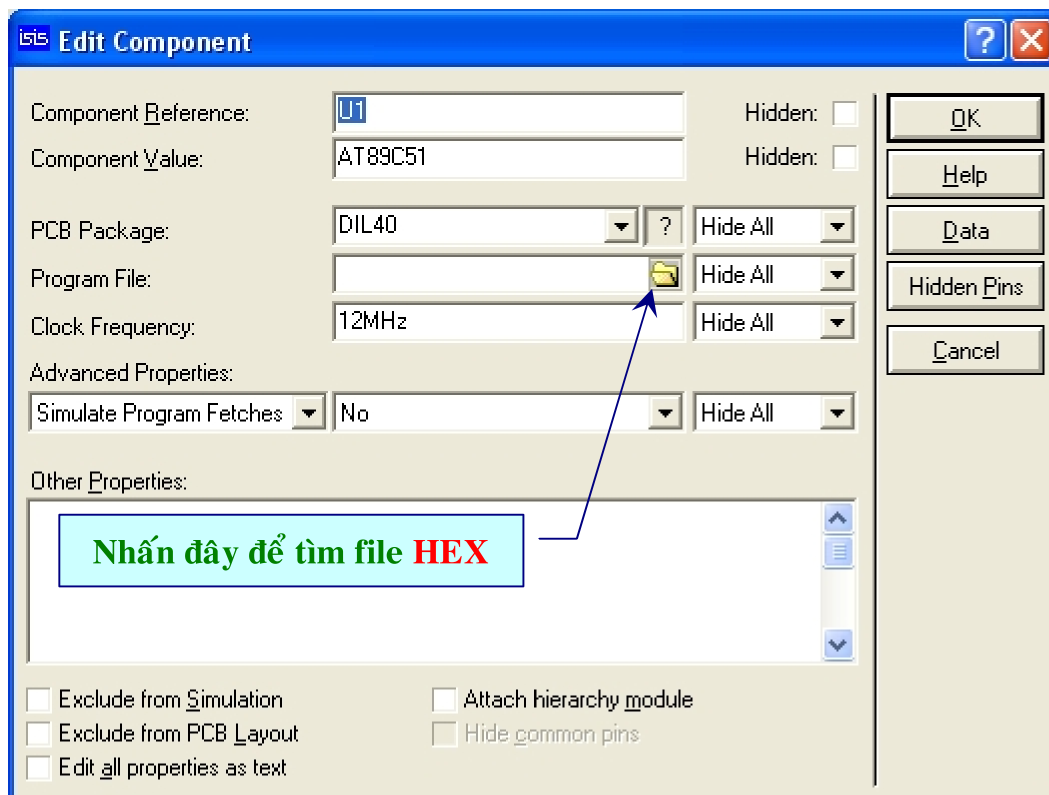
➢ ở cửa sổ vẽ mạch mô phỏng (***ISIS Professional***), chọn menu ***Source → Build All***. Một cửa sổ báo kết quả biên dịch sẽ xuất hiện cho biết biên dịch chương trình thành công hay thất bại.



Nếu biên dịch bị báo lỗi, ta mở file ***.LST** trong thư mục chứa file ***.ASM** để xem phần lỗi nằm ở đâu và chỉnh sửa trong chương trình theo các bước nhập chương trình ở trên.

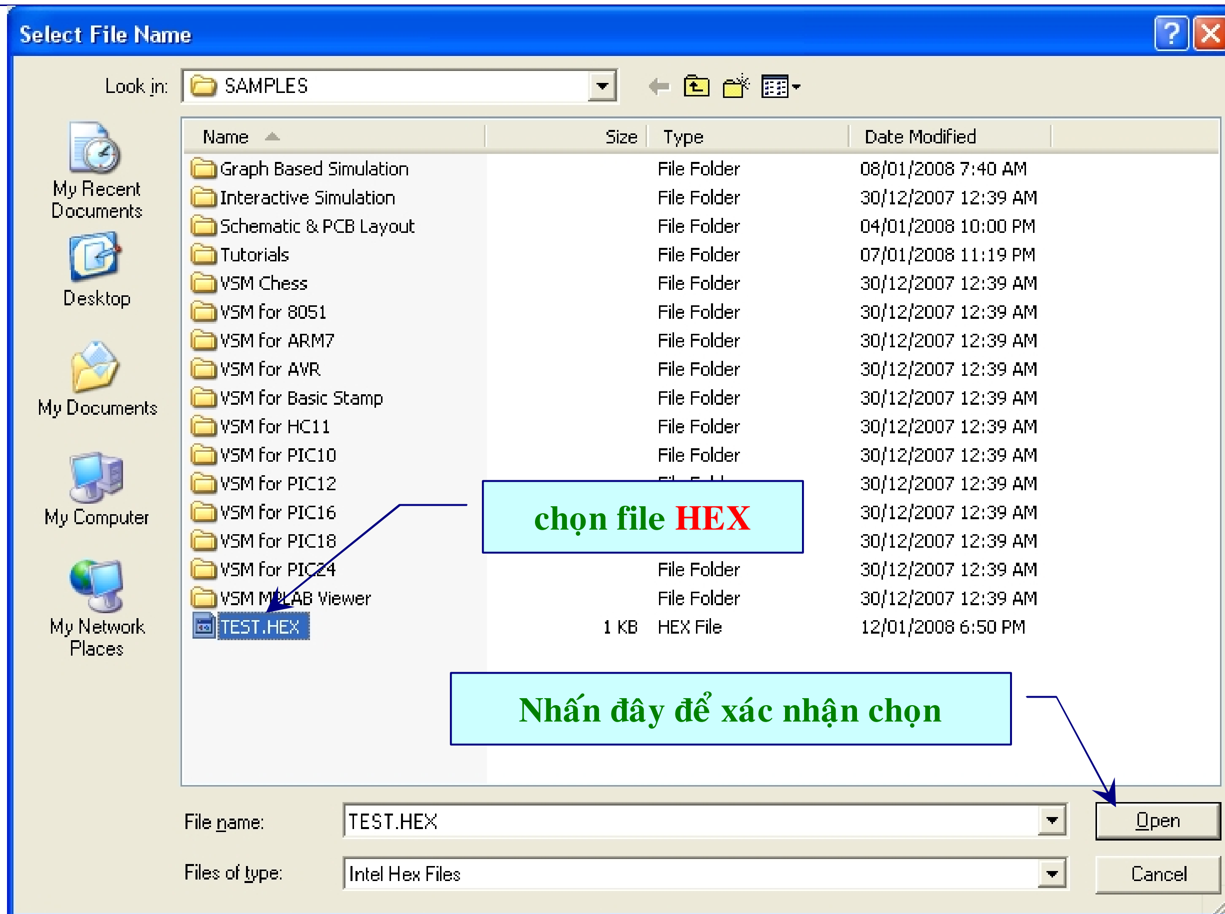
Nếu biên dịch ***OK***, ta đóng cửa sổ ***BUILD LOG*** và qua bước kế tiếp.

➢ **Click** phải (hoặc **double click**) lên IC AT89C51 để nạp chương trình cho phép chạy mô phỏng. Cửa sổ ***Edit Component*** sẽ xuất hiện.

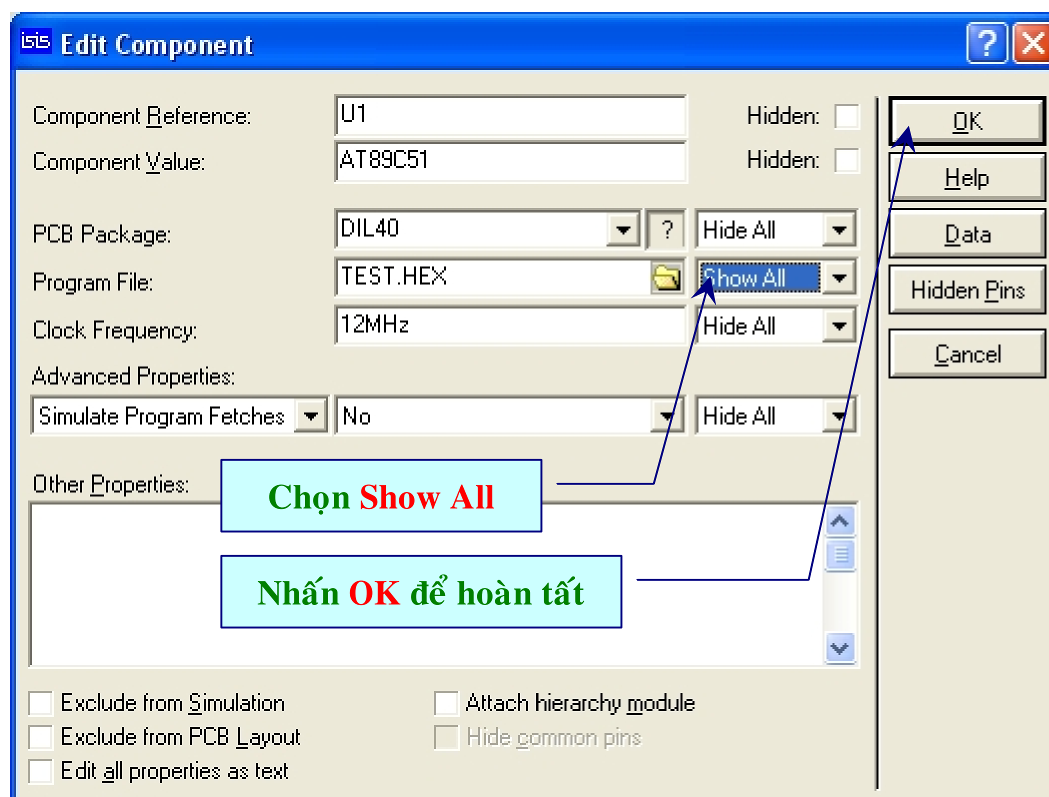


➢ Nhấn nút [biểu tượng thư mục] để chọn đường dẫn đến file chương trình đã biên dịch ***.HEX.** Hộp thoại ***Select file name*** sẽ xuất hiện**.**

➢ **Double click** lên file **HEX** để chọn file ***.HEX** cần mô phỏng để nạp cho vi điều khiển.

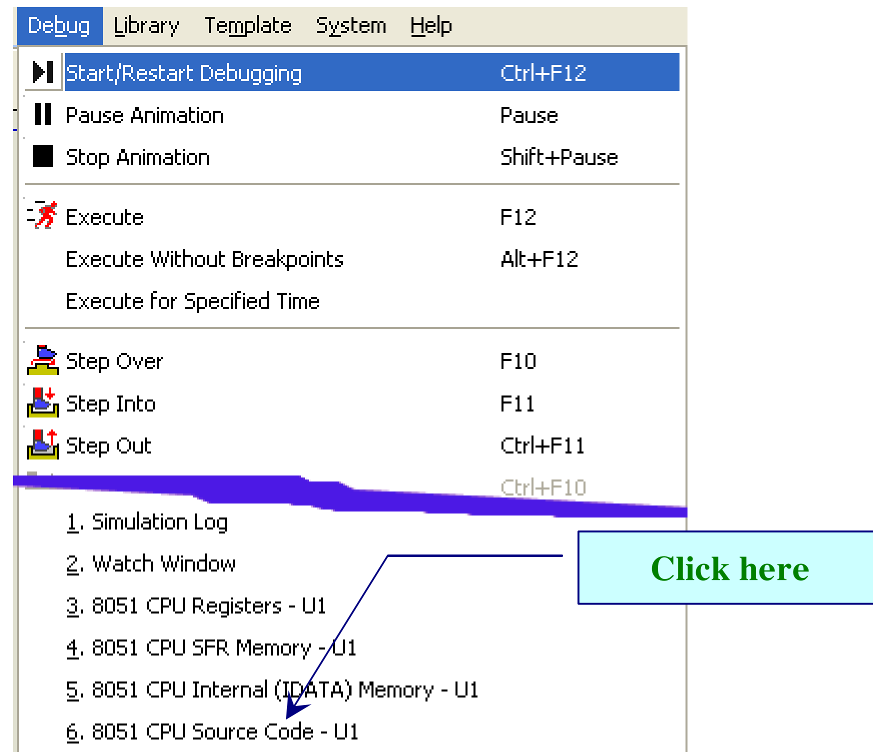


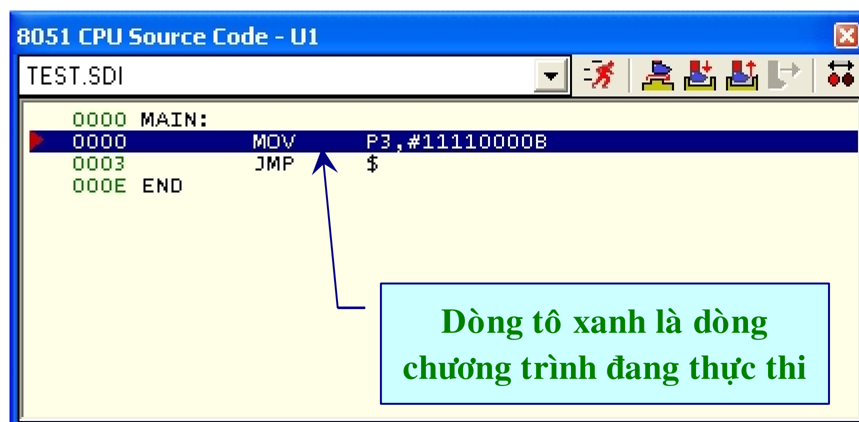
➢ Chọn ***Show All*** để cho phép khi chạy mô phỏng ta có thể xem chương trình đang thực thi đến đâu và kết quả hiển thị là kết quả của việc thực hiện bởi câu lệnh nào.



➢ Chọn menu ***Debug → Start/Restart debugging*** (hoặc nhấn **Ctrl + F12**). Phần mềm Proteus sẽ chạy bước mô phỏng đầu tiên rồi dừng lại chờ.

➢ Chọn menu ***Debug → 6. 8051 CPU Source Code – U1*** để mở cửa sổ theo dõi chương trình đang chạy.





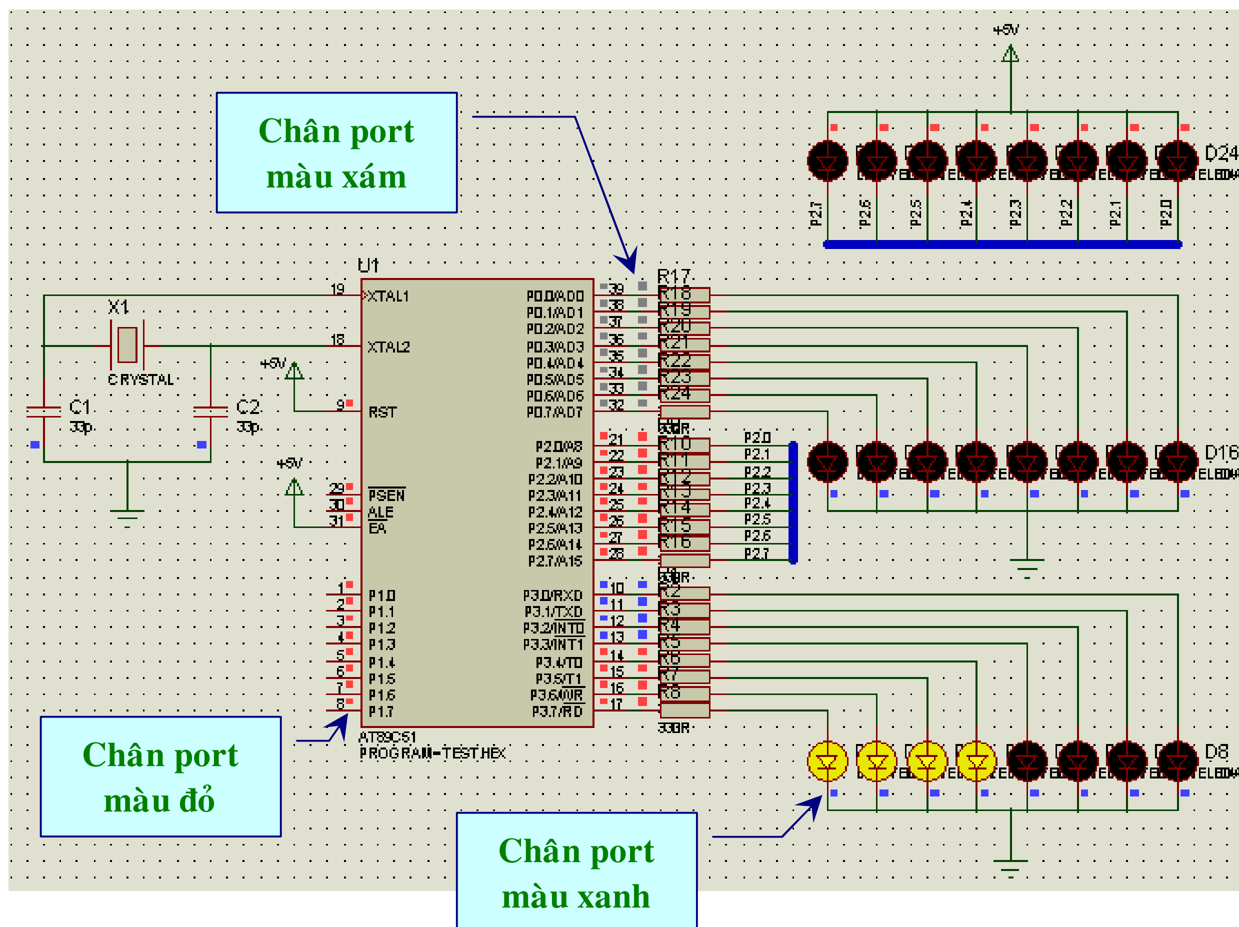
- Nút ***Execute*** (hoặc nhấn **F12**) – cho phép chạy chương trình mà không cần theo dõi nữa.
- Nút ***Step Over*** (hoặc nhấn **F10**) – cho phép chạy qua chương trình con mà không đi vào chi tiết của chương trình con này.
- Nút ***Step Into*** (hoặc nhấn **F11**) – cho phép chạy chi tiết bên trong chương trình con.
- Nút ***Step Out*** (hoặc nhấn **Ctrl + F11**) – cho phép thoát khỏi chương trình con để trở về cấp cao hơn.

Thường ta dùng nút ***Step Over*** để khảo sát hoạt động của mạch và chỉ dùng các nút khác khi muốn đi vào chi tiết chương trình.

Ngoài ra ta có thể sử dụng các nút thực hiện mô phỏng ở thanh nút điều khiển mô phỏng

- Nút – cho phép chạy chương trình mà không cần theo dõi.
- Nút – chạy từng bước chương trình.
- Nút – tạm ngừng việc thực thi chương trình.
- Nút – huỷ bỏ việc thực thi chương trình.

Sau đây là kết quả thực hiện chương trình mô phỏng:



Khi mô phỏng chạy, các đường mạch mặc định sẽ hiển thị trạng thái của nó. Ta thấy các đường mạch có 1 trong 3 màu là: màu **xanh dương**, màu **đỏ** hoặc màu **xám**.

- Màu **xanh dương** – đường mạch có điện áp ở mức thấp (mức logic 0).
- Màu **đỏ** – đường mạch có điện áp ở mức cao (mức logic 1).
- Màu **xám** – đường mạch có điện áp ở mức không xác định (có thể là dạng cực thu hở)

Trên đây là các thao tác cơ bản nhất để thực hiện một mạch mô phỏng hoạt động của vi điều khiển họ 8051 dùng phần mềm Proteus 7.1. Ngoài các thao tác trên, phần mềm Proteus còn cho phép thực hiện đo đạc tín hiệu dùng các công cụ đo đạc ảo như Volt kế, Ampere kế, Oscilloscope,... hoặc các máy phát tín hiệu xung clock, hàm mũ,... Các công cụ này có thể tìm thấy ở thanh công cụ mô phỏng.

- Nút **nguồn tín hiệu** - **click** chọn loại nguồn tín hiệu trong danh sách ***GENERATORS*** và **click** lên vùng trống của bản vẽ để đặt nguồn tín hiệu.
- Nút **Volt kế ảo** - **click** lên đường mạch cần đo để đặt Volt kế ảo.
- Nút **Ampere kế ảo** - **click** lên đường mạch ngay chân linh kiện cần đo để đặt ampere kế ảo.
- Nút **thiết bị ảo** - **click** chọn loại thiết bị đo ảo trong danh sách ***INSTRUMENTS*** và **click** lên vùng trống của bản vẽ để đặt thiết bị đo ảo.

Hướng dẫn biên dịch chương trình dùng phần mềm biên dịch khác

Có nhiều phần mềm để biên dịch một chương trình vi điều khiển. Tuy nhiên các phần mềm này được chia làm 2 loại dựa trên nền hệ điều hành được sử dụng: DOS hay Windows.

A. Biên dịch bằng công cụ DOS.

Trước khi biên dịch bằng công cụ DOS, chương trình vi điều khiển phải được viết trước dùng các phần mềm soạn thảo trong Windows như Notepad, Winword,... và lưu ở các định dạng text thông dụng như ***.TXT, *.A51, *.ASM**,... với phần tên file nên có độ dài không quá 8 ký tự và lưu file càng gần thư mục gốc càng tốt. File chương trình nên nằm cùng thư mục với chương trình biên dịch. Tốt nhất chúng ta nên sử dụng Notepad để soạn thảo.

Để biên dịch trong DOS, ta dùng 2 chương trình:

- ***ASM51.EXE*** để biên dịch chương trình sang dạng file object (***.OBJ**).
- ***OH.EXE*** để thực hiện chuyển các file object thành các file **HEX** (***.HEX**) cho chương trình nạp vi điều khiển có thể đọc được. Một số chương trình nạp vi điều khiển không có khả năng đọc file HEX, ta phải sử dụng thêm chương trình ***HEX2BIN.EXE*** để chuyển file HEX trên về dạng file BIN trước khi nạp.

Để biên dịch chương trình, ta thực hiện như sau:

- nhấn ***Start*** → ***Programs*** → ***Accessories*** → ***Command Prompt*** để mở cửa sổ DOS hoặc
- nhấn ***Start*** → ***Run...*** rồi gõ lệnh ***cmd*** để mở cửa sổ DOS.

Trong cửa sổ DOS, ta dùng các lệnh chuyển ổ đĩa và lệnh chuyển thư mục để di chuyển đến thư mục có chứa chương trình biên dịch. Như trong ví dụ là thư mục C:\ASM.

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\trungphan>cd c:\asm

C:\ASM>
```

Gõ lệnh dịch chương trình sang file Object:

ASM51 <tên file chương trình>

Chú ý: tên file chương trình phải ở dạng 8.3 của DOS.

Ví dụ: để dịch chương trình TEST.TXT sang dạng OBJ ta đánh lệnh như sau

ASM51 TEST.TXT

```
C:\WINDOWS\system32\cmd.exe

C:\ASM>ASM51 TEST.TXT

DOS 5.0 (038-N) MCS-51 MACRO ASSEMBLER, V2.2
Copyright 1979, 1983, 1986 Intel Corporation

ASSEMBLY COMPLETE, NO ERRORS FOUND
```

Trên màn hình sẽ thông báo cho ta biết là chương trình có bị lỗi hay không. Nếu có lỗi, ta mở file ***.LST*** để xem báo lỗi và quay lại chương trình ban đầu để sửa (như ví dụ là file ***TEST.LST***). Nếu chương trình không lỗi, ta tiếp tục biên dịch từ file Obj sang file HEX bằng cách gõ lệnh:

OH <tên file với phần mở rộng .OBJ>

Ví dụ:

```
C:\WINDOWS\system32\cmd.exe

C:\ASM>ASM51 TEST.TXT

DOS 5.0 (038-N) MCS-51 MACRO ASSEMBLER, V2.2
Copyright 1979, 1983, 1986 Intel Corporation

ASSEMBLY COMPLETE, NO ERRORS FOUND

C:\ASM>OH TEST.OBJ

DOS 5.0 (038-N)  OH  V1.0
Copyright 1986 Intel Corporation
```

Bây giờ ta có thể sử dụng một chương trình đi kèm mạch nạp để nạp cho vi điều khiển.

B. Biên dịch bằng công cụ WIN.

Có rất nhiều công cụ biên dịch trong Windows như

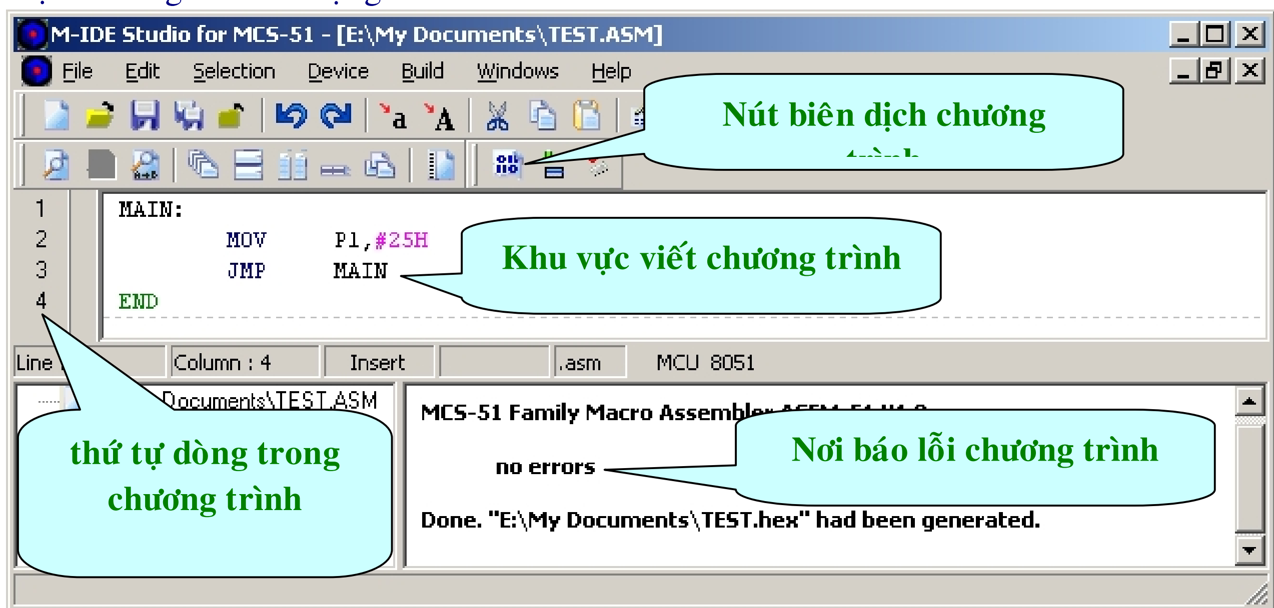
- **8051 IDE** (free trial download at www.acebus.com/win8051.htm)

- **M-IDE51** (free download at www.opcube.com/home.html)
- Keil (www.keil.com)
- …

Trong đó **M-IDE51** là công cụ sử dụng miễn phí. Các công cụ này cho phép chúng ta soạn thảo chương trình ngay trong phần mềm mà không cần các phần mềm soạn thảo khác. Ngoài ra, các phần mềm này còn có thể biên dịch các chương trình viết bằng ngôn ngữ ANSI C.

Sau khi cài đặt chương trình, ta **click** vào biểu tượng 

trên desktop hoặc chọn ***Start*** **➔** ***programs*** **➔** ***MIDE-51 package*** **➔** ***MIDE-51*** để chạy chương trình. Giao diện chương trình có dạng



Lưu ý:

Khi sử dụng phần mềm này, khi tạo mới một file chương trình (dùng menu ***File*** **➔** ***New***) ta phải **LẬP TỨC** lưu lại với tên file có phần mở rộng là "**.ASM**" (đánh đầy đủ cả phần mở rộng này) nếu không thì khi bấm nút biên dịch thì chương trình sẽ bị treo.

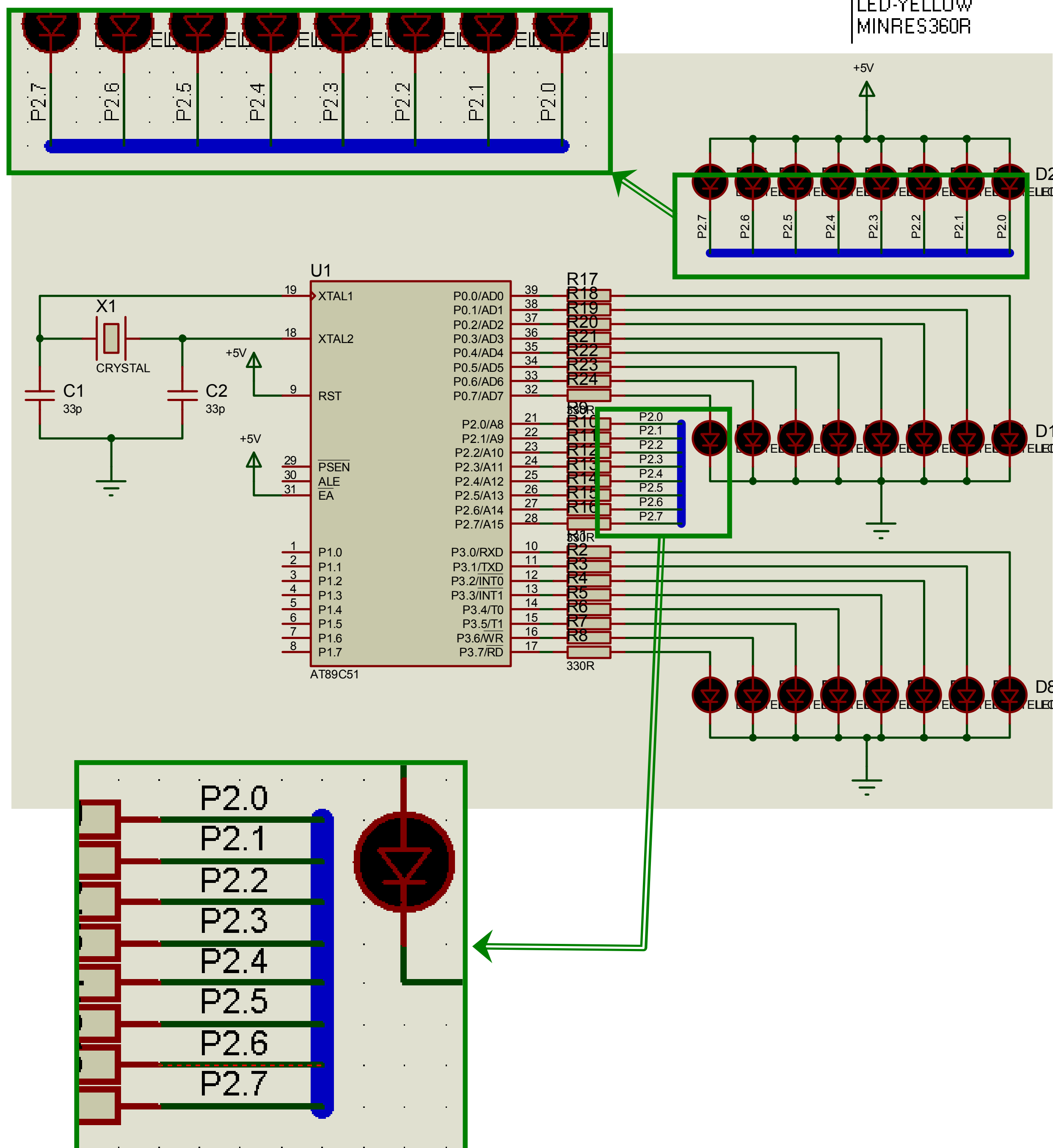
Ví dụ: nhập là **TEST.ASM** chứ không được nhập là **TEST** ở phần ***File name.***

BÀI 2

GIAO TIẾP VỚI LED ĐƠN

Mở chương trình ISIS và lấy các linh kiện trong danh sách như hình bên và thực hiện vẽ mạch sau.

P L DEVICES
AT89C51
AVX0402NP033P
CRYSTAL
LED-BLUE
LED-GREEN
LED-RED
LED-YELLOW
MINRES360R



Mở cửa sổ chương trình **MIDE–51** và nhập đoạn chương trình sau:

```
MAIN:
        MOV  P3,#00001111B
        JMP  MAIN
END
```

Biên dịch chương trình sang file **HEX** và mở cửa sổ ISIS để nạp chương trình trên cho AT89C51.

Chạy mô phỏng.

Nhiệm vụ.

- Quan sát hiện tượng xuất hiện trên mạch khi chạy mô phỏng.
- Nếu thay Port 3 trong chương trình bằng Port 2, quan sát hiện tượng xảy ra trên các led nối vào Port 2.
- Nếu thay Port 3 trong chương trình bằng Port 0, quan sát hiện tượng xảy ra trên các led nối vào Port 0. Hãy cho biết tại sao với cùng cách mắc led như Port 3 nhưng ở Port 0 led có hiện tượng như vậy? Hãy đề xuất biện pháp khắc phục.

  Gợi ý: xem đặc tính cấu tạo của chân Port 0.
- Hãy rút ra kết luận, cách mắc led như ở Port 3 và Port 2 là kiểu nào sau đây: CA (Common Anode – Anode chung), CC (Common Cathode – Cathode chung). Với từng kiểu mắc đó, hãy cho biết muốn 1 led nào đó sáng thì chân điều khiển tương ứng phải ở mức mấy (0 hay 1)? Nếu muốn tắt thì sao?

Bài tập.

Viết các đoạn chương trình thực hiện các yêu cầu sau:

1) Các led nối vào Port 3 sáng dần rồi tắt dần và lặp lại liên tục.

2) Các led nối vào Port 2 có 1 led sáng chạy từ trái qua phải rồi từ phải qua trái và lặp lại liên tục.

Hãy chèn thêm một câu lệnh gọi chương trình con **DELAY1S** (phải viết khai báo chương trình con **DELAY1S**) sau mỗi lần xuất dữ liệu ra Port.

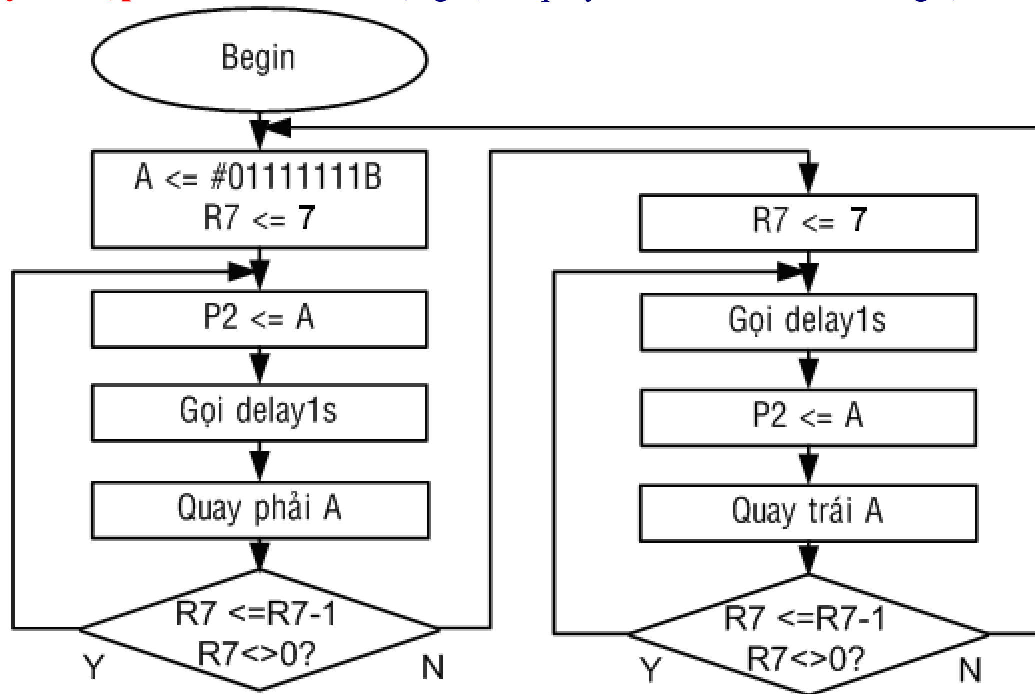
Hãy quan sát hiện tượng trên mạch khi có và không có chèn câu lệnh gọi **DELAY1S** ở trên.

Gợi ý chương trình con delay1s:

```
DELAY1S:
        PUSH    ACC
        MOV     TMOD,#01H
        MOV     A,#20
LOOP:
        MOV     TH0,#HIGH(–50000)
```

```
        MOV     TL0,#LOW(–50000)
        SETB    TR0
        JNB     TF0,$
        CLR     TR0
        CLR     TF0
        DJNZ    ACC,LOOP
        POP     ACC
RET
```

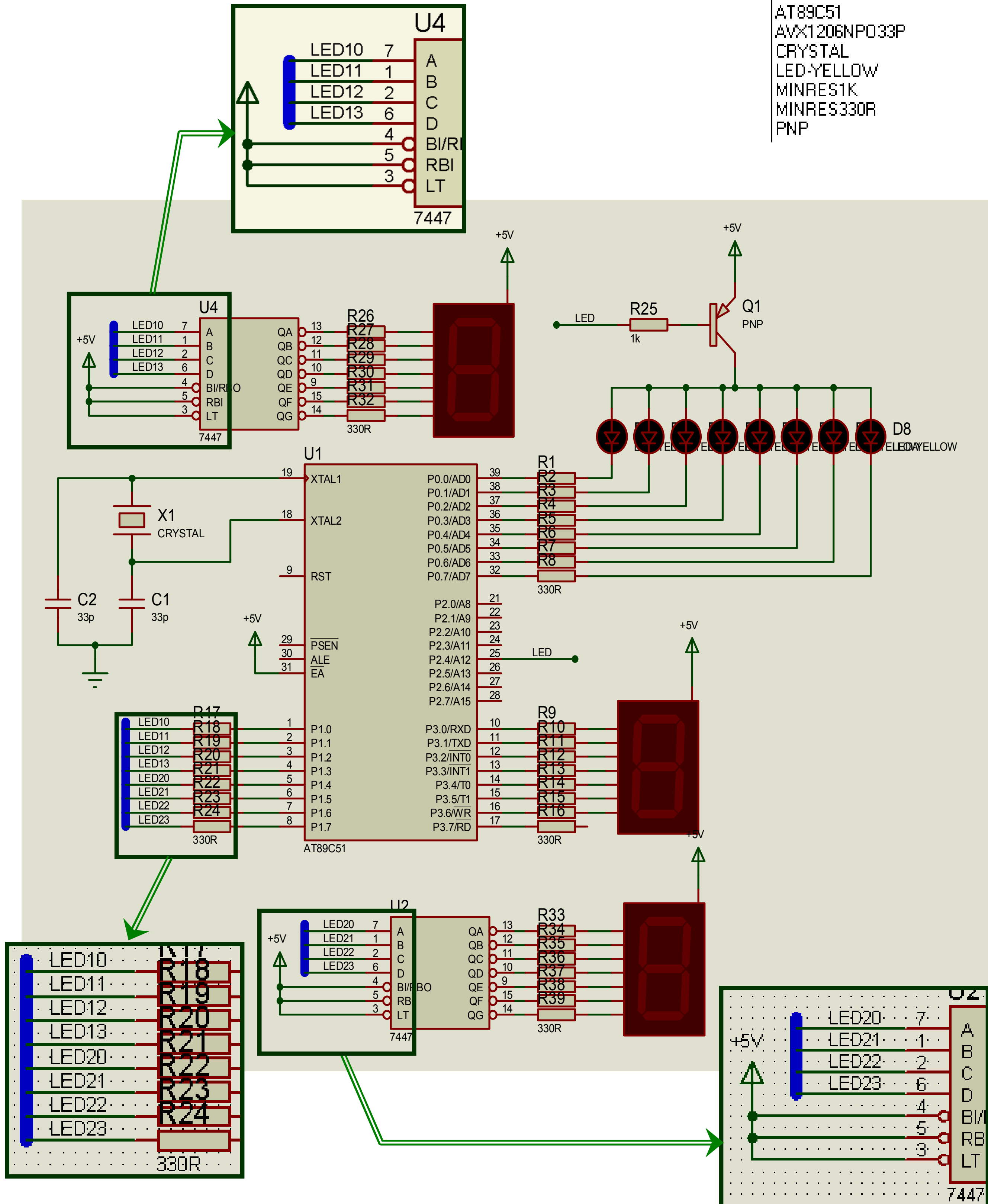
Gợi ý bài tập 2 – ta có thể sử dụng lệnh quay trái/ theo lưu đồ đề nghị như sau:



BÀI 3

GIAO TIẾP VỚI LED 7 ĐOẠN

Mở chương trình ISIS và lấy các linh kiện như trong danh sách rồi thực hiện vẽ mạch sau



Mở cửa sổ **MIDE–51** và nhập đoạn chương trình sau rồi thực hiện chạy mô phỏng mạch dùng **ISIS**.

```
MAIN:
        MOV  P1, #25H
        MOV  P0, #0AAH
        MOV  P3, #00010010B
        JMP  $
END
```

Nhiệm vụ.

- Quan sát hiện tượng xuất hiện trên mạch khi chạy mô phỏng.
- Hãy giải thích tại sao các led đơn nối vào Port 0 không sáng → đề xuất cách khắc phục bằng cách viết lại chương trình trên.

Gợi ý: quan sát/ đo đạc xem Transistor đã dẫn chưa? nếu chưa thì phải kích như thế nào để transistor dẫn được.

- Hãy cho biết phương thức giải mã led 7 đoạn của led 7 đoạn mắc vào port 3 và 2 led 7 đoạn mắc vào U2 và U4 là giải mã cứng hay giải mã mềm. Nếu ta thay **25H** trong chương trình bằng **25** thì điều gì xảy ra.
- Hãy quan sát cách xuất một số cụ thể lên các led 7 đoạn trong mạch.

Bài tập.

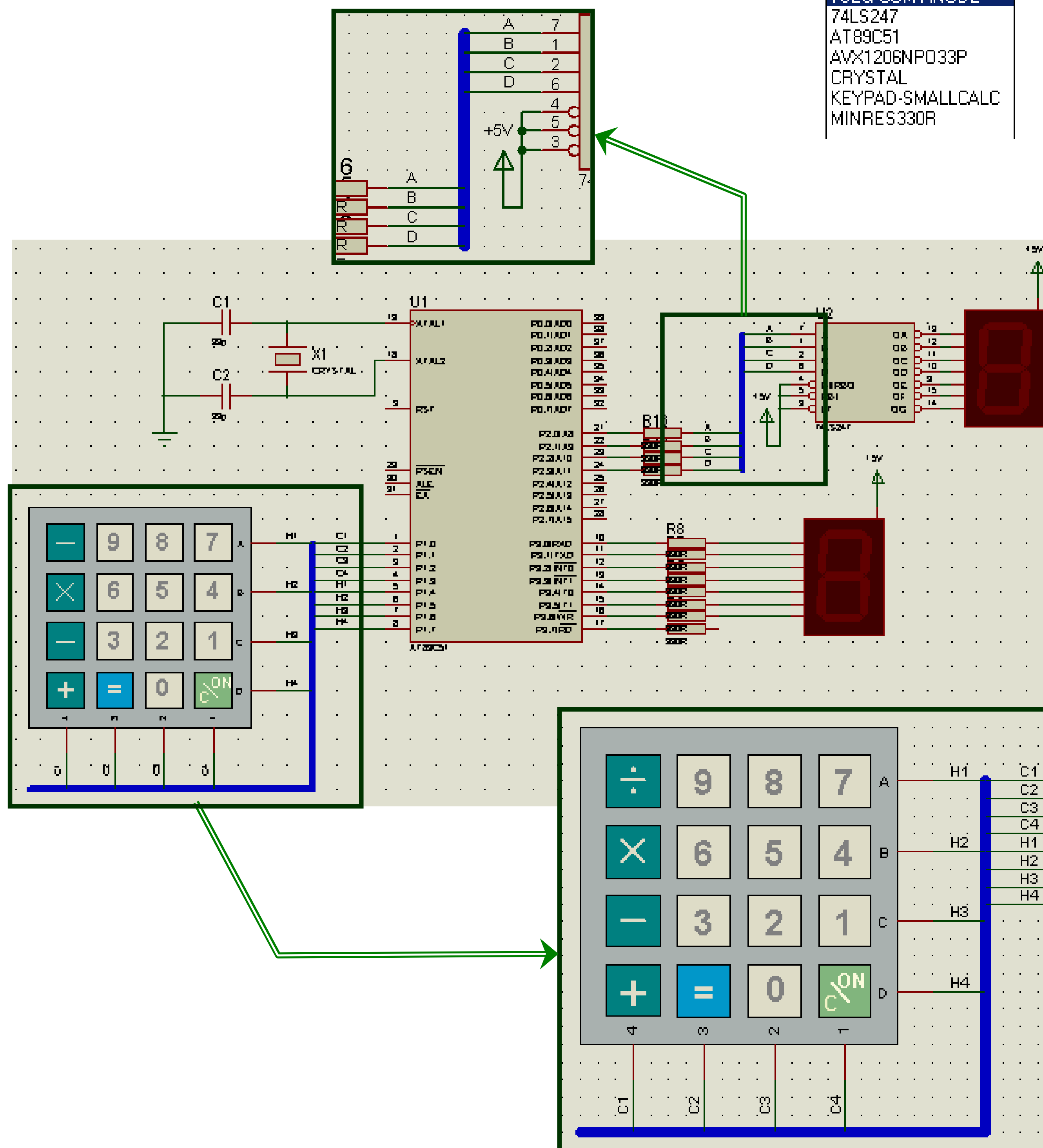
Viết các đoạn chương trình thực hiện các yêu cầu sau:

1) Hãy viết chương trình thực hiện đếm lên từ 0 → 9 trên led 7 đoạn ở port 3 và lặp lại liên tục.
2) Hãy viết chương trình thực hiện đếm xuống từ 9 → 0 trên led 7 đoạn ở port 3 và lặp lại liên tục.
3) Hãy viết chương trình thực hiện đếm lên từ 00 → 99 trên 2 led 7 đoạn ở port 1 và lặp lại liên tục.
4) Hãy viết chương trình thực hiện đếm xuống từ 99 → 00 trên 2 led 7 đoạn ở port 1 và lặp lại liên tục.
5) Hãy viết chương trình thực hiện đếm lên trên 2 led 7 đoạn ở port 1 từ 00 đến một số đặt trước (ví dụ: 56) thì chớp các led đơn ở port 0.

BÀI 4

GIAO TIẾP VỚI BÀN PHÍM HEXA

Mở chương trình ISIS và lấy các linh kiện như trong danh



Mở cửa sổ chương trình MIDE–51 và nhập đoạn lệnh sau rồi thực hiện mô phỏng dùng ISIS.

Chú ý:

- chọn nút ***component*** để có thể tác động lên các nút khi thực hiện mô phỏng trong ISIS
- Các ký hiệu trên bàn phím hexa chỉ là decal mặt nạ phím chứ không phải là giá trị thực của các nút nhấn.

Đoạn chương trình:

```
MAIN:
        CALL    GET_KEY
        JNC     MAIN
        MOV     P2,A
        JMP     MAIN
GET_KEY:
        ;xem phần gợi ý chương trình con get_key ở dưới
RET
END
```

Gợi ý Chương trình con quét phím:

Đây là gợi ý về chương trình con quét phím không chống rung.

Nếu kết thúc chương trình con mà C = 0 → không có phím nhấn, ngược lại nếu C=1 → có phím nhấn và giá trị phím nhấn nằm trong thanh ghi A

```
GET_KEY:
        PUSH    05H         ; STORE REGISTERS
        PUSH    06H
        PUSH    07H

        MOV     A,#0FEH     ; START WITH COLUMN 0
        MOV     R6,#4  ; USE R6 AS COUNTER
TEST:
        MOV     P1,A        ; ACTIVE COLUMN LINE
        MOV     R7,A        ; SAVE ACC
        MOV     A,P1        ; READ BACK PORT 0
        ANL     A,#0F0H     ; ISOLATE ROW LINES
        CJNE    A,#0F0H,KEY_HIT ; ROW LINE ACTIVE?
        MOV     A,R7        ; NO → MOVE TO NEXT
        RL      A           ; COLUMN LINE
        DJNZ    R6,TEST
```

```
        CLR     C           ; NO KEYPRESSED
        SJMP    EXIT        ; EXIT WITH C = 0
KEY_HIT:
        MOV     R7,A        ; SAVE IN R6
        MOV     A,#4        ; PREPAIR FOR CALCULATE
        CLR     C           ; COLUMN WEIGHTING
        SUBB    A,R6        ; 4 – R6 = WEIGHTING
        MOV     R6,A        ; SAVE IN R6
        MOV     A,R7        ; RESTORE SCAN CODE
        SWAP    A           ; PUT IN LOW NIBBLE
        MOV     R5,#4  ; USE R5 AS COUNTER
AGAIN:
        RRC     A           ; ROTATE UNTIL 0
        JNC     DONE        ; DONE WITH C = 0
        INC     R6          ; ADD 4 UNTIL ACTIVE ROW
        INC     R6
        INC     R6
        INC     R6
        DJNZ    R5,AGAIN
DONE:
        SETB    C           ; C=1 IF KEYPRESSED
        MOV     A,R6        ; CODE IN A
EXIT:

POP     07H         ; RESTORE REGISTER
POP     06H
POP     05H
RET
```

Gợi ý Chương trình con quét phím có chống rung:

Đây là gợi ý về chương trình con quét phím có chống rung, nếu có phím nhấn, giá trị phím nhấn sẽ cất trong stack, đến khi phím được nhả, giá trị phím nhấn được trả về trong thanh ghi A. Khi sử dụng đoạn chương trình này, ta không cần xét đến cờ C nữa vì chương trình chỉ thoát ra khi nhả phím và giá trị phím nhấn trả về trong thanh ghi A.

```
DEBOUNCED:
        MOV     R3,#50; debounce count
BACK:
```

```
        CALL    GET_KEY     ; key pressed?
        JNC     DEBOUNCED ; no: check again
        DJNZ    R3,BACK   ; yes: repeat checking 50 times
        PUSH    ACC         ; store key code in stack
BACK2:
        MOV     R3,#50; wait gor key up
BACK3:
        CALL    GET_KEY     ; key pressed?
        JC      BACK2       ; yes: keep checking
        DJNZ    R3,BACK3    ; no: repeat checking 50 times
        POP     ACC         ; restore key code
RET
```

Chọn nút ***component*** để có thể tác động lên các nút khi thực hiện mô phỏng trong ISIS

Nhiệm vụ.

- Dùng chuột nhấn giữ một nút trên bàn phím, quan sát xem led có hiện số không? Cho biết giá trị mà phím nhấn đó trả về trong thanh ghi A.
- Nếu bỏ lệnh **JNC MAIN** thì điều gì xảy ra?
- Nếu đảo thứ tự dây nối cột trong mạch, điều gì xảy ra khi mô phỏng.
- Nếu đảo thứ tự dây nối hàng trong mạch, điêu gì xảy ra khi mô phỏng.
- Thay đoạn chương trình chính trên bằng đoạn chương trình sau rồi thực hiện lại các bước trên.

```
MAIN:
        CALL DEBOUNCED
        MOV  P2,A
        JMP  MAIN
GET_KEY:
        ;xem phần gợi ý chương trình con get_key ở trên
RET

DEBOUNCED:
        ;xem phần gợi ý chương trình con debounced ở trên
RET

END
```

Bài tập.

Hãy viết chương trình thực hiện các yêu cầu sau:

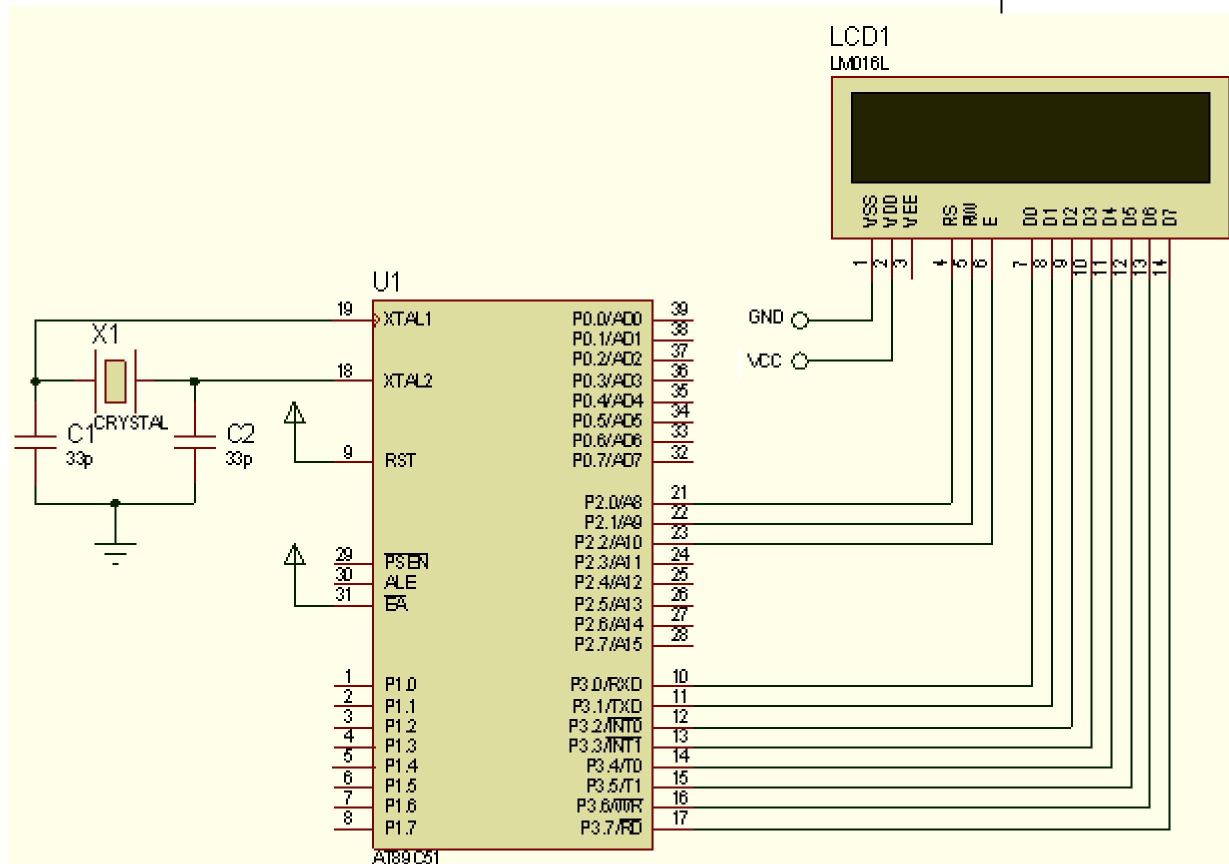
1) Nhập một số từ bàn phím và hiển thị số đó ra led ở Port 3 (gợi ý: dùng thêm MOVC).
2) Nhập một số từ bàn phím và thực hiện việc đếm lên trên led ở port 2 từ 0 đến số vừa nhập.
3) viết chương trình chuyển đổi bố trí bàn phím hexa chuẩn về dạng bố trí phím như trong sơ đồ mạch.
4) Vẽ thêm một mạch giải mã cứng led nối vào nibble cao của port 2, viết chương trình thực hiện yêu cầu sau:
5) Khi ta nhấn phím X, giá trị phím X sẽ hiện lên led ở nibble thấp. nếu nhấn tiếp phím Y thì giá trị phím X sẽ hiện lên led ở nibble cao và giá trị phím Y sẽ hiện lên led ở nibble thấp (gợi ý: dùng thêm SWAP).
6) viết tiếp chương trình ở bài 4 trên sao cho khi nhấn phím >9 thì thực hiện việc đếm lên từ 00 đến 2 số vừa nhập ở trên.

BÀI 5

GIAO TIẾP VỚI MÀN HÌNH LCD 16X2

Mở chương trình ISIS, lấy các linh kiện như trong danh sách và thực hiện vẽ mạch sau

P	L	DEVICES
AT89C51		
AVX1206NP033P		
CRYSTAL		
LM016L		



Mở cửa sổ chương trình MIDE–51 và nhập đoạn lệnh sau rồi thực hiện mô phỏng dùng ISIS:

```
; khai bao cac chan dieu khien
        RS      BIT     P2.0    ; register select
        RW      BIT     P2.1    ; Read or write
        E       BIT     P2.2    ; Enable
        DAT     EQU     P3      ; data port

        X       EQU     7Fh     ; X=dong 0->1
        Y       EQU     7Eh     ; Y=cot 0->15
;delay 20ms cho on dinh nguon dien
        CALL    delay20ms
;khoi dong LCD
```

```
        CLR     E
        CALL    init_lcd
;cac thao tac xu ly chinh
main:
        MOV     X,#0
        MOV     Y,#3
        MOV     DPTR,#chuoi1
        CALL    gotoXY
        CALL    out_string

        MOV     X,#1
        MOV     Y,#1
        CALL    gotoXY
        MOV     DPTR,#chuoi2
        CALL    out_string

        SJMP    $
        JMP     main
;---------------------------------------------
delay20ms:
        MOV     R6,#50
loop1:
        MOV     R7,#200
        DJNZ    R7,$
        DJNZ    R6,loop1
RET
;---------------------------------------------
delay40us:
        MOV     R7,#20
        DJNZ    R7,$
RET
;---------------------------------------------
gotoXY:
        ; set dia chi con tro hien thi den toa do (X,Y)
        ;dong 0: macot = 0-> 0fh
        ;dong 1: macot = 40h -> 4fh
```

```
        MOV     A,X
        MOV     B,#40H
        MUL     AB
        ADD     A,Y
        ORL     A,#80h

        CLR     RS
        CLR     RW
        MOV     DAT,A
        SETB    E
        CLR     E
        CALL    delay40us
RET
;---------------------------------------------
init_lcd:
        ; function set command
        CLR     RS
        CLR     RW
        MOV     DAT,#00111000B
        SETB    E
        CLR     E
        CALL    delay40us

        ; display ON command
        CLR     RS
        CLR     RW
        MOV     DAT,#00001110B
        SETB    E
        CLR     E
        CALL    delay40us
RET
;---------------------------------------------
out_char:
        SETB    RS
        CLR     RW
        MOV     DAT,A                   ;A chua ky tu can xuat
        SETB    E
```

```
        CLR     E
        CALL    delay40us
RET
;-------------------------------------------
out_string:
;DPTR tro den chuoi can xuat, chuoi phai ket thuc bang so 0
        CLR     A
        MOVC    A,@A+DPTR
        JZ      end_of_string
        CALL    out_char
        INC     DPTR
        SJMP    out_string
end_of_string:
RET
;-------------------------------------------
chuoi1:
        DB      'TN Vi xu ly',0
chuoi2:
        DB      'Lop CDT306.1+2',0
END
```

Sau đây là một số thông số về LCD 16x2:

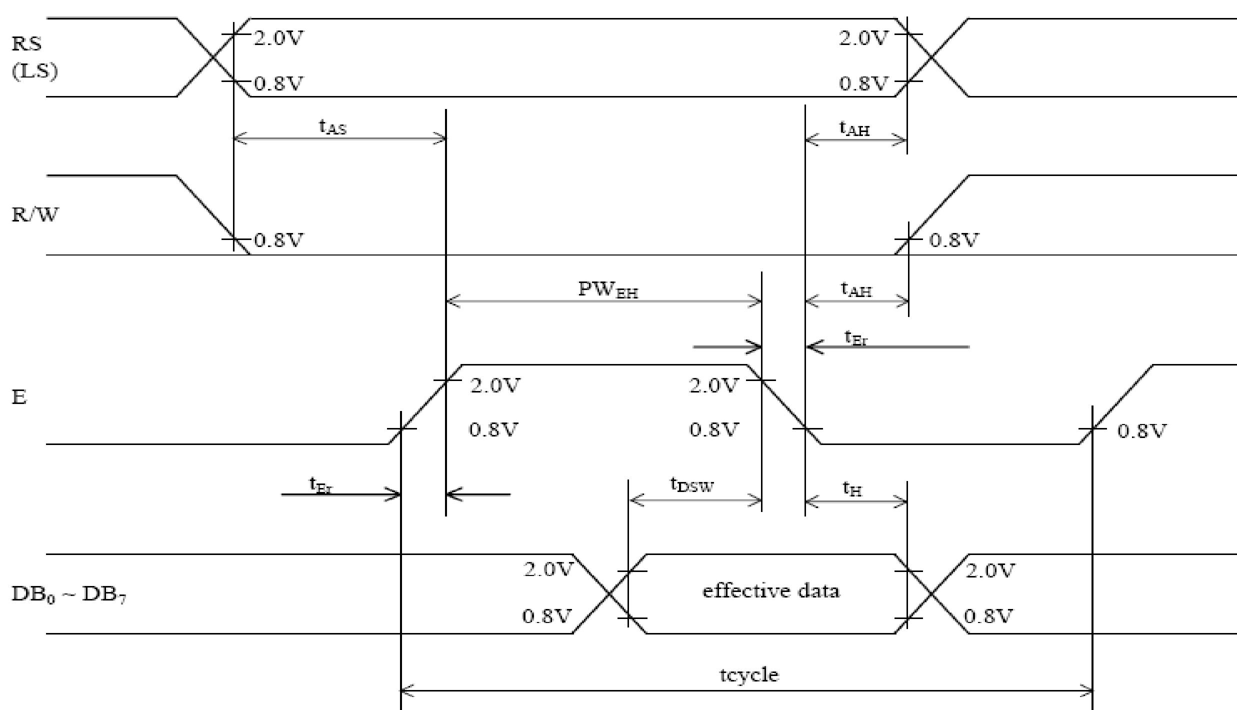
1. Mã tọa độ của con trỏ.

Column

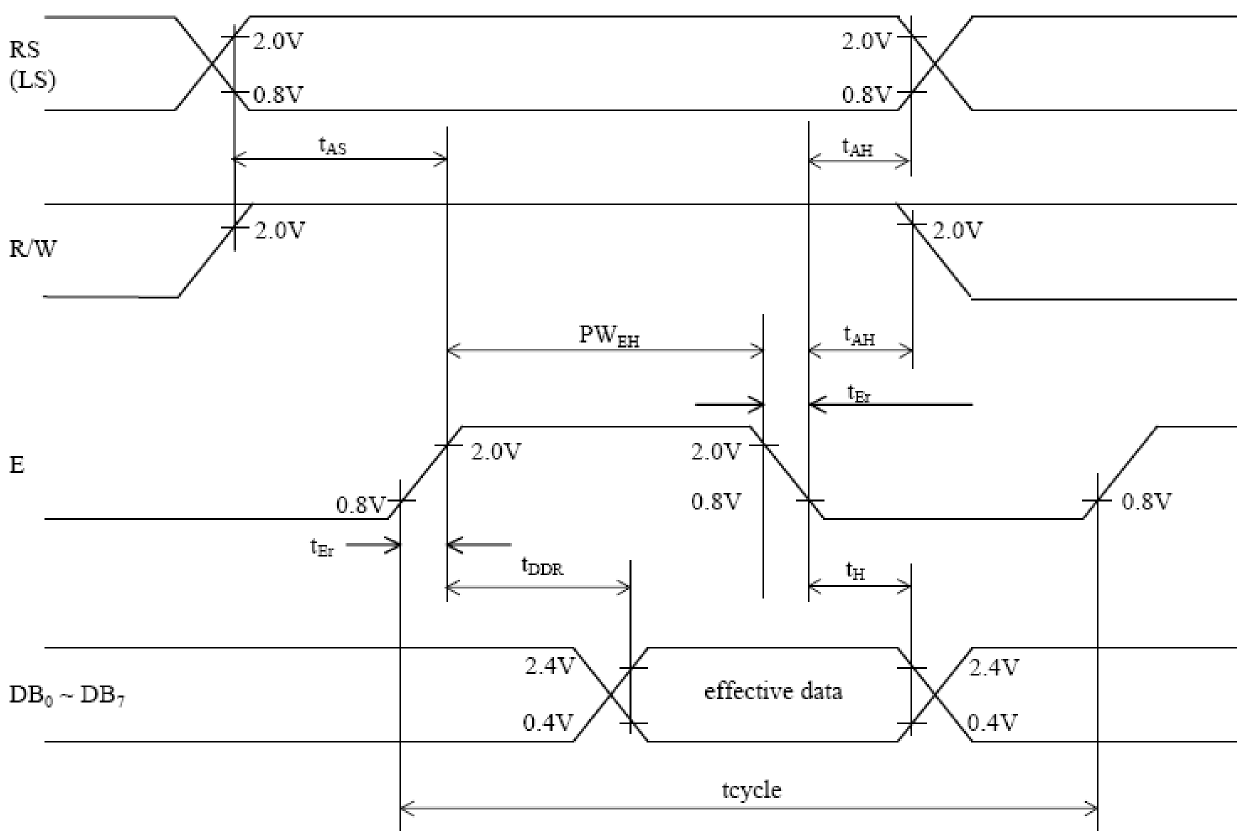
	1	2	3	. . .	15	16	. . .	19	20	. . .	31	32	. . .	38	39	40
line 1	00	01	02		0E	0F		12	13		1E	1F		25	26	27
line 2	40	41	42		4E	4F		52	53		5E	5F		65	66	67

16 x 2

2. giản đồ xung ghi data đến LCD.



3. giản đồ xung đọc data từ LCD.



BẢNG TẬP LỆNH ĐIỀU KHIỂN LCD

Instruction	Code										Description	Execution time (max.) when fcp or fosc is 250 kHz
	RS	R/W	DB_7	DB_6	DB_5	DB_4	DB_3	DB_2	DB_1	DB_0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter.	15.2ms
Return Home	0	0	0	0	0	0	0	0	1	x	Sets DD RAM address 0 in address counter. Also returns shifted display to original position. DD RAM contents remain unchanged.	15.2ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies shift or display. These operations are performed during data write and read.	40µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Sets ON/OFF of entire display (D), cursor ON/OFF (C), and blink of cursor position character (B).	40µs
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Moves cursor and shifts display without changing DD RAM contents.	40µs
Function Set	0	0	0	0	1	DL	N	F	x	x	Sets interface data length (DL), number of display lines (N) and character font (F).	40µs
Set CG RAM Address	0	0	0	1	ACG						Sets CG RAM address. CG RAM data is sent and received after this setting.	40µs
Set DD RAM Address	0	0	1	ADD							Sets DD RAM address. DD RAM data is sent and received after this setting.	40µs
Read Busy Flag & Address	0	1	BF	AC							Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	40µs
Write Data to CG or DD RAM	1	0	Write Data								Writes data into DD RAM or CG RAM.	40µs
Read Data from CG or DD RAM	1	1	Read Data								Reads data from DD RAM or CG RAM.	40µs
	I/D=1 : Increment I/D=0 : Decrement S=1 : Accompanies display shift S/C=1 : Display shift S/C=0 : Cursor move R/L=1 : Shift to the right R/L=0 : Shifts to the left DL=1 : 8 bits, DL=0 : 4 bits N=1 : 2 lines, N=0 : 1 line F=1 : 5x10 dots, F=0 : 5x7 dots BF=1 : Internally operating BF=0 : Can accept instruction										DD RAM : Display Data RAM CG RAM : Character Generator RAM ACG : CG RAM address ADD : DD RAM address. Corresponds to cursor address. AC : Address counter used for both DD and CG RAM address.	Execution time changes when frequency changes. Example: When fcp or fosc is 270kHz: 40µs x 250/270 = 37 µs

BẢNG MÃ CHỮ

High order bit / Low order bit	0000	0010	0011	0100	0101	0110	0111	1010	1011	1100	1101	1110	1111
X X X X0000	CG RAM (1)		0	@	P	`	p		ー	タ	ミ	α	p
X X X X0001	(2)	!	1	A	Q	a	q	。	ア	チ	ム	ä	q
X X X X0010	(3)	"	2	B	R	b	r	「	イ	ツ	メ	β	θ
X X X X0011	(4)	#	3	C	S	c	s	」	ウ	テ	モ	ε	∞
X X X X0100	(5)	$	4	D	T	d	t	、	エ	ト	ヤ	μ	Ω
X X X X0101	(6)	%	5	E	U	e	u	・	オ	ナ	ユ	σ	ü
X X X X0110	(7)	&	6	F	V	f	v	ヲ	カ	ニ	ヨ	ρ	Σ
X X X X0111	(8)	'	7	G	W	g	w	ァ	キ	ヌ	ラ	g	π
X X X X1000	(1)	(	8	H	X	h	x	ィ	ク	ネ	リ	√	x̄
X X X X1001	(2)	)	9	I	Y	i	y	ゥ	ケ	ノ	ル	⁻¹	y
X X X X1010	(3)	*	:	J	Z	j	z	ェ	コ	ハ	レ	j	千
X X X X1011	(4)	+	;	K	[	k	{	ォ	サ	ヒ	ロ	ˣ	万
X X X X1100	(5)	,	<	L	¥	l	|	ャ	シ	フ	ワ	¢	円
	(6)	-	=	M	]	m	}	ュ	ス	ヘ	ン	£	÷
	(7)	.	>	N	^	n	→	ョ	セ	ホ	゛	ñ	
	(8)	/	?	O	_	o	←	ッ	ソ	マ	゜	ö	█

Nhiệm vụ.

Hãy vẽ lại lưu đồ giải thuật của ví dụ trên.

Bài tập.

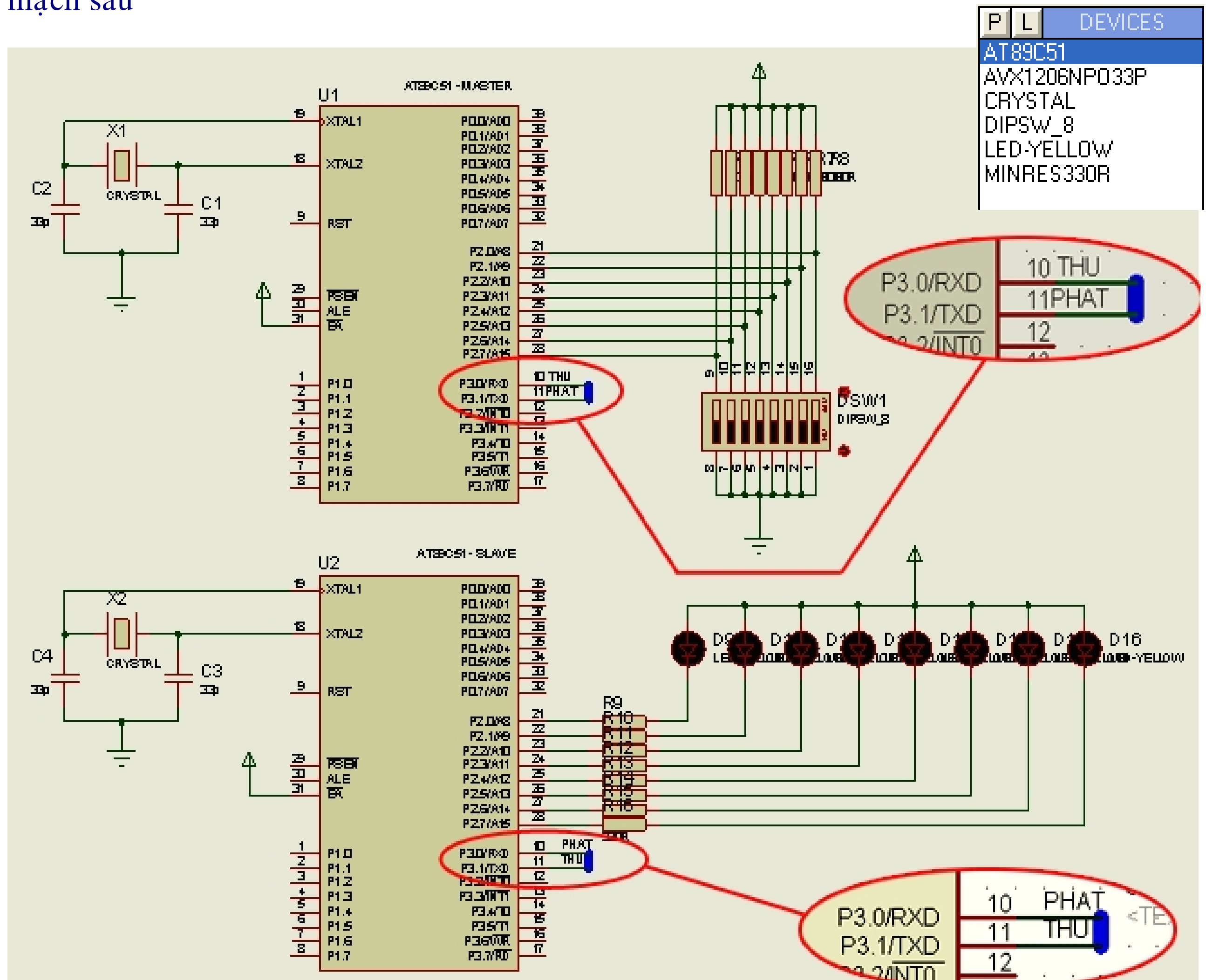
Hãy viết chương trình thực hiện các yêu cầu sau:

1) Xuất một dòng chữ xoay tròn trên dòng 1 của LCD.
2) Xuất nguyên một chuỗi ký tự chạy từ phải sang trái trên dòng 1 của LCD. Sau khi chuỗi ở dòng 1 hiển thị xong thì xuất nguyên chuỗi ký tự chạy từ trái sang phải ở dòng 2.
3) Xuất từng ký tự chạy từ phải sang trái trên dòng 1 LCD , các ký tự này ghép lại thành chuỗi trên dòng 1 LCD. Sau đó, xuất từng ký tự chạy từ trái sang phải trên dòng 2 của LCD, các ký tự này cũng ghép lại thành chuỗi trên dòng 2 của LCD.
4) Hiện một chuỗi ký tự trên dòng 1 của LCD. Sau đó, cho rơi từng chữ ở dòng 1 (từ trái sang phải) xuống dòng 2 LCD và cũng hợp thành chuỗi như ở dòng 1.

BÀI 6

TRUYỀN THÔNG NỐI TIẾP

Mở chương trình ISIS, lấy các linh kiện như trong danh sách và thực hiện vẽ mạch sau



Mở cửa sổ MIDE–51 và nhập đoạn chương trình sau rồi lưu vào đĩa với tên ***MASTER.ASM***, biên dịch chương trình này ra file ***MASTER.HEX***.

```
        MOV     SCON,#01010010B
        MOV     TMOD,#20H
        MOV     TH1,#-3
        MOV     TL1,#-3
        SETB    TR1
MAIN:
        CALL    READ_KEY
        CALL    PHAT
        JMP     MAIN
```

```
READ_KEY:
        MOV     A,P2
        RET
PHAT:
        JNB     TI,$
        CLR     TI
        MOV     SBUF,A
        RET
END
```

Mở một cửa sổ khác và nhập đọan chương trình sau rồi lưu vào đĩa với tên ***SLAVE.ASM***, biên dịch chương trình này ra file ***SLAVE.HEX***.

```
        MOV     SCON,#01010010B
        MOV     TMOD,#20H
        MOV     TH1,#-3
        MOV     TL1,#-3
        SETB    TR1
MAIN:
        CALL    THU
        CALL    XUAT_LED
        JMP     MAIN
XUAT_LED:
        MOV     P2,A
        RET
THU:
        JNB     RI,$
        CLR     RI
        MOV     A,SBUF
        RET
END
```

Mở cửa sổ chương trình ISIS, nạp chương trình ***MASTER.HEX*** cho con ***MASTER*** (U1) và nạp chương trình ***SLAVE.HEX*** cho con ***SLAVE*** (U2).

Nhiệm vụ.

- Chọn nút component để có thể tác động lên SW1 trong quá trình mô phỏng.
- Khi tác động các khóa của SW1, đèn có sáng đúng thứ tự như khóa không?
- Nếu không, hãy đề xuất cách khắc phục hiện tượng này.

Bài tập.

Hãy viết chương trình (cho cả MASTER và SLAVE) thực hiện các yêu cầu sau (có thể thiết kế lại mạch cho phù hợp với yêu cầu của đề bài).

1) SW1 đóng vai trò là bàn phím nhập các số cho MASTER. Khi MASTER nhận được các số này sẽ phát sang cho SLAVE. Nếu SLAVE nhận được số:
   - Sáng đèn tương ứng với số nhận được (đèn đánh số từ 0 → 7)
   - 13: sẽ chớp đèn liên tục cho đến khi nhận được giá trị khác.
   - Các số khác thì đảo giá trị hiện tại trên đèn.
2) Viết chương trình cho MASTER phát liên tục một ký tự chọn trước với tốc độ baud chọn trước(baudrate = 1200 → 19200 bps) . Giả sử SLAVE chưa biết tốc độ baud của MASTER, viết chương trình phát hiện tốc độ baud này. Khi MASTER giao tiếp được với SLAVE, thực hiện lại chức năng của đọan ví dụ.

BÀI 7

MỘT SỐ LƯU ĐỒ GIẢI THUẬT GỢI Ý